Suivi temps-réel d'objets 3D pour la réalité augmentée

Lucie Masson

N° d'ordre : D.U. 1625
EDSPIC : 342

UNIVERSITÉ BLAISE PASCAL - CLERMONT-FERRAND II

École Doctorale
Sciences Pour l'Ingénieur de Clermont-Ferrand

Thèse présentée par :
Lucie MASSON

Formation Doctorale IPIM :
Informatique, Productique et Imagerie Médicale

en vue de l'obtention du grade de

DOCTEUR D'UNIVERSITÉ

spécialité : Informatique

Suivi temps-réel d'objets 3D pour la réalité augmentée

Soutenue publiquement le 09 décembre 2005 devant le jury :

Mme Marie-Odile BERGER	Rapporteur et Examinateur
M. Vincent CHARVILLAT	Invité
M. Pierre COURTELLEMONT	Examinateur
M. Michel DHOME	Examinateur
M. Frédéric JURIE	Directeur de thèse
M. Augustin LUX	Président du jury
M. Éric MARCHAND	Rapporteur et Examinateur

Résumé

Ce mémoire de thèse a pour sujet le suivi temps réel d'objets en trois dimensions, dans le but de réaliser des applications de réalité augmentée. En effet la réalité augmentée nécessite des algorithmes de suivi stables et précis. Si l'on désire en plus que le suivi soit effectué en temps réel vidéo, il faut alors trouver des compromis entre précision des résultats et vitesse de traitement.

Ce mémoire contient la description des trois algorithmes de suivi développés durant cette thèse. Ils illustrent le cheminement suivi par nos travaux durant ces trois années, c'est-à-dire le suivi d'objets de plus en plus complexes, d'abord planaires, puis simples objets 3D, et enfin objets 3D complexes de modèle 3D inconnu.

Le premier algorithme permet de suivre des objets planaires peu texturés. Il s'agit d'une extension d'un algorithme de suivi de plans efficace et rapide, basé sur l'utilisation d'informations de texture, auquel nous avons ajouté une composante de suivi de contour afin de pouvoir l'utiliser sur un ensemble plus vaste de motifs.

Une fois ce travail sur le suivi planaire effectué, nous avons adapté l'algorithme de suivi de textures au suivi d'objets en trois dimensions. En utilisant de multiples occurrences de cet algorithme, réparties sur la surface de l'objet à suivre, couplées à un algorithme itératif d'estimation de pose, nous sommes parvenus à suivre en temps réel des objets simples effectuant des translations et des rotations à 360 degrés.

Cet algorithme étant limité par le fait qu'il nous faut connaître un modèle 3D de l'objet à suivre, nous avons ensuite cherché à réaliser un algorithme permettant, lors d'une phase d'apprentissage, de générer un modèle statistique de l'objet à partir de vues clefs 2D. Basé sur le même algorithme de suivi de texture que précédemment, cet algorithme ne détermine pas la pose 3D de l'objet suivi mais décrit sa position comme étant la déformation d'une grille 2D.

Abstract

This thesis aims at real time tracking of 3D objects, in order to develop augmented reality applications. Augmented reality needs a stable and accurate tracking. Adding that we also want real time video tracking, we had to found compromises between results's precision and speed of treatment.

This report contains the description of three tracking algorithms developed during this thesis. They illustrate the progression accomplished by our work during these three years, which is the tracking of increasingly complex objects, first planar objects, then simple 3D objects, and finally complex 3D objects with an unknown model.

With the first algorithm we were able to track few textured planar objects. It is an extension of a fast and accurate algorithm that tracks textured plans. We added a contour component to this algorithm to make it track a greater amount of patterns.

When this work on planar tracking have been done, we adapted the texture tracking algorithm to 3D objects. Using multiple occurrences of this algorithm distributed on the object's surface to follow, and associating them with an iterative algorithm of pose estimation, we managed to track in real time some simple objects during their translations and rotations up to 360 degrees.

This algorithm was limited by the fact that a 3D model of the followed object had to be known. That's why we tried to find an algorithm that would be able, during an off-line learning stage, to generate a statistical model of the object from key-views. Based on the same texture algorithm that the one used in previous algorithms, it doesn't estimate the objects' pose, but characterizes it by a 2D spline deformation.

Table des matières

Introduction **1**

1 État de l'art **7**
1.1 Introduction 7
1.2 La réalité augmentée 7
1.2.1 Les dispositifs de réalité augmentée 9
1.2.2 Affichage des éléments virtuels 11
1.2.3 Capteurs de suivi 12
1.2.4 Calibration 15
1.3 Le suivi temps réel 15
1.3.1 Introduction 15
1.3.2 Principes généraux 16
1.3.3 Détection d'objets 19
1.3.4 Estimation des paramètres de mouvement 20
1.3.5 Suivi par association de données 21
1.4 Suivi planaire de motifs texturés 26
1.4.1 Algorithme 27
1.4.2 Une application : jeu de labyrinthe en réalité augmentée . 31

2 Suivi planaire contour/texture **35**
2.1 Introduction 35
2.1.1 Suivi de contour / suivi de texture 35
2.1.2 Contribution et plan du chapitre 36
2.2 Une méthode hybride pour le suivi de motif 36
2.2.1 Représentation du motif 37
2.2.2 Représentation du mouvement 38
2.2.3 Suivi du motif 40
2.3 Les phases d'apprentissage et de suivi 42
2.3.1 Une mesure relative du vecteur de forme 42

2.3.2 Apprentissage 43
2.3.3 Suivi 44
2.4 Validation expérimentale 45
2.5 Conclusion 50

3 Suivi rigide avec modèle 3D 55
3.1 Introduction 55
3.2 Algorithme de suivi 3D développé 55
3.3 Appariements 3D-2D 57
3.4 Estimation de la pose de l'objet 59
3.4.1 Algorithme théorique 62
3.4.2 Algorithme pratique 67
3.5 Résultats expérimentaux 69
3.6 Conclusion et perspectives 70

4 Suivi sans modèle 3D 75
4.1 Introduction 75
4.2 État de l'art relatif à cette méthode 76
4.3 Description de l'algorithme développé 77
4.3.1 Apprentissage du modèle de déformations 78
4.3.2 Apprentissage pour le suivi de petites vignettes 86
4.3.3 Algorithme de suivi déformable 86
4.4 Résultats 89
4.4.1 Suivi de visage 89
4.4.2 Suivi d'objet déformable 90
4.5 Conclusions et perspectives 92

5 Conclusion et perspectives 93
5.1 Contribution de notre travail 93
5.2 Perspectives 94
5.2.1 Optimisation et répartition des traitements 94
5.2.2 Amélioration de la technique de suivi planaire 95
5.2.3 Remplacement de la technique de suivi planaire 95

Bibliographie 103

Table des figures

1 Croissance de la puissance de calcul des ordinateurs, par Ray Kurzweil (cette image est sous licence « Creative Commons Attribution License v.1.0 »). 3

1.1 Réalité augmentée/Réalité virtuelle 8
1.2 Visiocasques - haut : miroir transparent - bas : miroir opaque . . . 10
1.3 Le tableau magique 11
1.4 Gestion des occultations en réalité augmentée. La vache brune, l'arbre rond et la voiture ont été ajoutés. (images tirées du site de Vincent Lepetit) 12
1.5 Détermination des paramètres d'illumination - gauche : image de départ - milieu : insertion d'une balle sans respect des paramètres d'illumination - droite : avec respect de l'illumination(d'après le site de Jürgen Stauder) 13
1.6 Dispositif de réalité augmentée de Newman *et al.* [55] 14
1.7 Décomposition d'une séquence vidéo 16
1.8 suivi : principe général 18
1.9 Suivi d'une main par méthode de Comaniciu *et al.* [13] 23
1.10 Suivi d'un visage par méthode de Jepson *et al.* [33] 24
1.11 Suivi d'un livre par méthode de Lowe [47] 25
1.12 Suivi d'un objet de modèle CAO connu par Drummond et Cipolla [23] 26
1.13 Sélection de points - gauche : sélection d'un point par baquet - droite : voisinage des points sélectionnés 28
1.14 Phase d'apprentissage pour le suivi plan. 31
1.15 Dispositif de réalité augmentée : jeu du labyrinthe 32
1.16 Images de labyrinthe - haut : bille virtuelle dans le labyrinthe - bas : dispositif physique 33

2.1 Distance du point au contour le plus proche 38

2.2 Repère image / repère motif 39
2.3 Exemples de suivi de motifs texturés (1) 47
2.4 Exemples de suivi de motifs texturés (2) 48
2.5 Séquence de test générée artificiellement (images 5 et 15) 51
2.6 Estimation des coordonnées du coin supérieur droit du motif par les différents algorithmes 52
2.7 Zones de convergence pour des translations suivant x et y. haut : objet suivi, gauche : uniquement des textures, droite : uniquement des points de contour, bas : méthode hybride 53

3.1 Exemple de répartition des patches à la surface d'un cube, répartition en fonction des points d'intérêt. 57
3.2 Les vignettes, des algorithmes de suivi plans à quatre degrés de liberté 58
3.3 Précision du suivi 59
3.4 Séquence de test de suivi des vignettes 60
3.5 Domaine de convergence des vignettes 61
3.6 Algorithme itératif de Lowe : plans d'interprétation 63
3.7 Convergence de l'algorithme itératif de Lowe pour quatre calculs de pose différents. 65
3.8 Convergence de l'algorithme itératif de Lowe pour quatre calculs de pose différents (suite). 66
3.9 Suivi d'un cube occulté (1) 70
3.10 Suivi d'un cube occulté (2) 71
3.11 Médiane des résidus, en pixels, pour une séquence artificielle . . . 72
3.12 Estimation d'une translation sur l'axe Y 73
3.13 Extraits d'une séquence réelle de suivi 74

4.1 Déformation des splines après une légère rotation de d'une canette. haut : point d'intérêt appariés suivant SIFT - bas : estimation de la déformation de la grille 79
4.2 Différence de séparation entre le scree-test et et la conservation de 85% de la variance - marqueur bleu : séparation par variance, marqueur vert, séparation par scree-test 84
4.3 Modèle de canette - haut gauche : vue clef de l'objet - haut droite et bas gauche : déformation de l'image clef suivant l'une des composantes de déformation obtenue par ACP - bas droite : exagération de la déformation 85
4.4 Modèle de visage 87

4.5 Patches sélectionnés sur une image de chaussure. gauche : la grille - droite : avec des vignettes sur chaque nœud 88
4.6 Ajustement de la grille par TPS. haut : sans régularisation - bas :avec une régularisation itérative 90
4.7 Suivi de visage humain 91
4.8 Substitution et déformations de visages. 91
4.9 Suivi d'un tapis de souris « mou » 92

Liste des tableaux

2.1 Algorithme de suivi hybride 46
2.2 Temps d'exécution suivant le type de suivi utilisé 49

4.1 Algorithme d'ajustement TPS itératif entre deux images 81
4.2 Algorithme du *scree-test* 83
4.3 Nombre de valeurs propres conservées par scree-test et par séparation par variance 83
4.4 Fonction de coût pour l'optimisation par Levenberg-Marquardt . . 89

Introduction

Les travaux décrits dans le présent document appartiennent au domaine de la vision par ordinateur, domaine qui consiste à réaliser des algorithmes capables de traiter des images ou des séquences d'images, et de résoudre des problèmes tels que reconnaître un visage, suivre un objet, se localiser dans l'espace…

Il s'agit principalement d'actions que les êtres humains, et certains animaux, sont capables d'accomplir. Le plus souvent, on aimerait pouvoir faire accomplir à des machines des tâches qui nécessitent l'utilisation de la vision pour pouvoir être réalisées correctement.

Mais bien que l'un des buts de la vision par ordinateur, ou vision artificielle, soit de parvenir à faire réaliser à des machines ce que l'œil humain fait naturellement, les recherches en vision ne cherchent pas pour autant à copier le fonctionnement de l'œil.

En effet même si le but à atteindre peut sembler identique à ce que fait un œil naturel, il n'est pas toujours pertinent de vouloir imiter la nature sous forme d'algorithme. Ainsi, dans un autre domaine, on a vu apparaître des programmes d'échec capables de rivaliser avec les meilleurs joueurs mondiaux. Pourtant leurs algorithmes ne reflètent pas la façon de jouer d'un être humain. Un bon joueur va utiliser son expérience, des connaissances acquises sur la stratégie et la tactique, un peu de lecture des coups à venir. Mais une machine n'acquiert pas d'expérience, ou du moins elle n'est pas programmée pour accomplir cette tâche, et le concept de stratégie n'a pas de sens pour elle. Par contre elle possède une mémoire parfaite, et est capable de lire de façon exhaustive des séquences de coups très longues.

Alors qu'un joueur humain va directement s'intéresser aux quelques coups intéressants de la partie à un moment donné, réfléchir aux séquences auxquels ils mènent, et faire son choix, le joueur artificiel va tester tous les coups possibles, même les plus stupides, regarder toutes les séquences, et choisir celle qui, finalement, est la meilleure d'après ses algorithmes. Les deux raisonnent différemment, mais tous deux jouent des coups logiques pouvant mener à la victoire.

Lors de la fameuse confrontation entre Kasparov et Deep Blue, des observa-

teurs avaient même comparés certains coups de Deep Blue à ceux d'un « Grand Maître ».

La problématique est la même en vision par ordinateur. L'homme et la machine n'ont pas les mêmes points forts en matière de vision, ni les mêmes défauts.

Ainsi, l'image fournie par un « œil artificiel », comme une caméra, est parfaite : elle correspond exactement à l'environnement tel qu'il est. L'œil humain, lui, construit une image mentale de l'environnement parfois très différente de la réalité. L'œil ayant un champ de vision nette très faible, il effectue des mouvements rapides et c'est le cerveau qui se charge de construire une image complète, ce qui amène à certaines déficiences étonnantes.

Une étude de Daniel Simons et Christopher Chabris, de l'Université de Havard, a ainsi démontré que des gens occupés à compter le nombre de passes entre des joueurs de basket étaient incapables de percevoir la présence d'un individu déguisé en gorille passant au milieu des joueurs et esquissant un pas de danse [63].

D'un autre côté, le duo œil/cerveau a des capacités d'analyse qu'il est difficile d'obtenir artificiellement. Un être humain ou un primate évolué est capable, à partir d'images d'un objet prises sous différents angles, de reconnaître cet objet vu sous un angle inédit. Comment imiter artificiellement cette faculté de retrouver la forme en trois dimensions d'un objet à partir d'images en deux dimensions, et de retrouver des éléments cachés ?

L'histoire de la vision par ordinateur commence par celle du traitement d'image. Des algorithmes de traitement sont étudiés dès le début du 20^{e} siècle, et c'est en 1920 que Harry Bartholomew et Maynard McFarlane effectuent la première numérisation d'images afin de pouvoir transmettre des photos de journaux entre l'Europe et l'Amérique par câble sous-marin. Les images en noir et blanc sont codées pour la transmission par câble, puis décodées de l'autre côté de l'Atlantique. Avec cette méthode il ne faut que trois heures pour envoyer une image, au lieu de la semaine nécessaire auparavant.

Le traitement d'image est une extension du traitement de signal, où l'image est considérée comme un signal à deux dimensions. Ce domaine profite de la redécouverte, en 1965, de l'algorithme de transformée de Fourier rapide (FFT) par James Cooley et John Tuckey [18], qui avait en fait été inventé dès 1805 par Carl Friedrich Gauss. Les algorithmes de traitement d'image sont des outils de base de la vision par ordinateur.

Les avancées en matière de vision ont été directement liées aux avancées en informatique : lorsque les ordinateurs possédaient peu de mémoire, il était impossible de charger la totalité d'une image en mémoire pour la traiter, et des travaux de recherche étaient même réalisés sur le traitement d'image ligne par ligne !

C'est pour cette raison qu'il n'y a eu que peu d'avancées entre 1920 et 1960,

et que la vision par ordinateur prend son essor dans les années 1980, époque à partir de laquelle les ordinateurs deviennent suffisamment puissants pour pouvoir commencer à réaliser des traitements efficaces (voir figure 1).

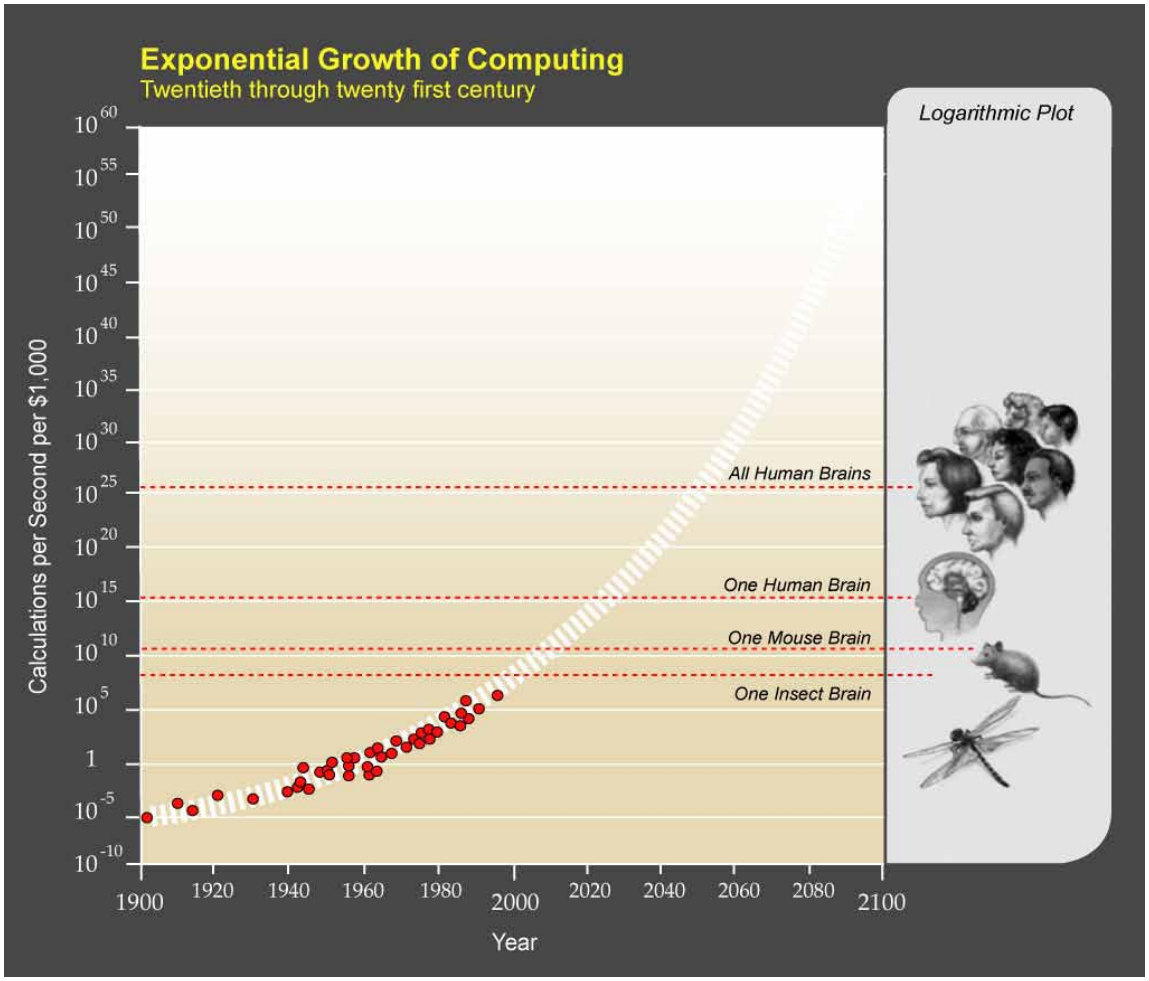


FIG. 1 – Croissance de la puissance de calcul des ordinateurs, par Ray Kurzweil (cette image est sous licence « Creative Commons Attribution License v.1.0 »).

Les années 1970 sont la période de l'essort des systèmes experts dans le domaine de l'intelligence artificielle. Il s'agit de programmes informatiques visant à simuler le comportement d'un expert humain dans un domaine très précis tels que les diagnostiques médicaux ou les échanges boursiers.

Un système expert est composé d'une base de connaissance, et d'un moteur d'inférence. Les connaissances se composent de faits et de règles (du type

« si… alors… »), et le moteur d'inférence combine « faits » et « règles » pour créer de nouveaux « faits » et parvenir à une solution au problème posé.

Les systèmes experts sont alors la base de l'intelligence artificielle, ils sont appliqués à de nombreux domaines, et sont même utilisés dans l'industrie.

Mais l'application des systèmes experts à la vision par ordinateur est un échec, car il s'avère impossible de créer des bases de données permettant d'alimenter des systèmes « simples » de reconnaissance de formes. La complexité algorithmique de ces systèmes est trop importante, et il faudra encore plusieurs années pour voir apparaître des algorithmes de vision accomplissant des tâches comparables à celles de l'œil humain.

Dans les années 1970 apparaissent aussi les premiers algorithmes classiques du traitement d'image : segmentation, détection de contours et de lignes,…

Et c'est dans les années 1980, avec le développement de l'informatique grand public, que l'intelligence artificielle, et par extension la vision par ordinateur, se développent véritablement.

En particulier les travaux de David Marr [52] en neurosciences modifient considérablement la façon d'envisager la vision artificielle. Pour David Marr, la compréhension du cerveau humain doit se faire en observant les problèmes que parvient à résoudre le cerveau, et les solutions qu'il leur trouve.

Plutôt que de produire des systèmes experts, avec leurs bases de connaissances, la recherche s'oriente alors vers la résolution de problèmes de vision plus simples, et qui permettent en plus d'obtenir des informations sur le fonctionnement de la vision humaine.

Parmi ces problèmes de vision se trouve celui du suivi d'éléments visuels, qui sera traité dans ce document. Suivre un « objet », connu ou non, d'une image à l'autre dans une séquence vidéo a en effet bon nombre d'applications, par exemple dans le domaine médical ou la vidéo-surveillance. L'une de ces applications est la *réalité augmentée*, c'est-à-dire l'ajout dans une vidéo d'éléments non présents initialement.

Le but du présent document est de décrire trois méthodes relatives au suivi par vision, utiles pour réaliser des applications de réalité augmentée. Le suivi doit donc être précis et rapide, afin que les éléments insérés apparaissent de façon fluide et cohérente dans les images.

Ces trois méthodes sont basées sur un algorithme de suivi basé sur la texture décrit dans le premier chapitre. Chaque méthode est un peu plus complexe que la précédente, et permet d'assouplir les hypothèses initiales sur les conditions à remplir pour que le suivi soit possible.

La structure de ce document est la suivante : tout d'abord, dans un premier chapitre, nous faisons l'état de l'art de la réalité augmentée ainsi que du suivi

par vision. Les trois chapitres suivants décrivent chacun une méthode de suivi développée durant cette thèse.

La première, au chapitre deux, décrit le suivi d'objets planaires à l'aide d'un mélange d'informations de texture et de contour.

La deuxième, au chapitre trois, décrit un algorithme robuste de suivi d'objets simples en trois dimensions, réalisé à partir de multiples instances de l'algorithme de suivi décrit au chapitre un. Cette méthode nécessite de connaître parfaitement, *a priori*, le modèle 3D de l'objet suivi.

Le quatrième chapitre décrit une méthode de suivi qui reprend l'algorithme du chapitre trois, mais où la connaissance *a priori* du modèle n'est plus nécessaire car celui-ci est généré lors d'une phase d'apprentissage.

Chapitre 1

État de l'art

1.1 Introduction

Ce chapitre a pour but, après une description de ce qu'est la réalité augmentée, de faire un inventaire de ce qui existe dans ce domaine ainsi que dans le domaine du suivi par vision.

Après avoir montré les différents dispositifs de réalité augmentée, et constaté la nécessité des algorithmes de suivi par vision dans ces travaux, nous présenterons les principaux algorithmes de suivi existants. Enfin, dans une dernière partie, nous développerons de façon plus précise un algorithme de suivi de plans texturés qui sera la base des travaux présentés dans les autres chapitres de cette thèse.

1.2 La réalité augmentée

On appelle *réalité augmentée* l'insertion d'éléments virtuels dans une séquence vidéo « réelle », c'est-à-dire une séquence dont les images sont issues directement d'une caméra telle qu'il en existe dans le commerce. A la différence de la *réalité virtuelle*, où la totalité de la scène est générée artificiellement, une séquence en *réalité augmentée* contient à la fois des éléments réels et virtuels.

R. Azuma [4] définit la réalité augmentée comme ayant les propriétés suivantes :

- elle combine les éléments réels et virtuels dans un environnement réel ;
- elle est interactive, et par conséquent fonctionne en temps réel ;
- elle cherche à ajuster des objets réels et virtuels les uns par rapport aux autres.

La réalité augmentée consiste le plus souvent à ajouter des objets virtuels dans

un environnement réel, et donc à « augmenter » notre vision, mais il ne faut pas la limiter à cela : elle peut au contraire consister à retirer des objets réels dans des images, par exemple dans le cas de simulations architecturales pour simuler le retrait d'un immeuble, ou pour pouvoir insérer les éléments du futur projet à la place d'éléments existants. De plus, potentiellement, cela ne concerne pas uniquement la vue, mais aussi l'ouïe, l'odorat, le toucher.

Il s'agit donc, plus que d'ajouter des éléments à la réalité, de modifier cette réalité.

D'après Milgram [53] il y existe une continuité entre les environnements réels et virtuels, que l'on peut modéliser par la figure 1.1 Il n'y a pas de réelle rup-

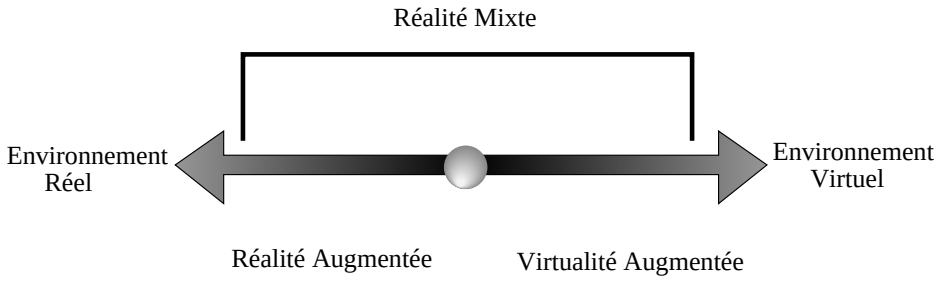


FIG. 1.1 – Réalité augmentée/Réalité virtuelle

ture entre la virtualité augmentée, qui consiste à ajouter des éléments réels dans un environnement virtuel, et la réalité augmentée. Toutes deux appartiennent au domaine de la « réalité mixte », mélange d'éléments réels et virtuels.

Le concept de réalité augmentée date des années 60 et des travaux de Sutherland [65], qui présentait en 1968 un visiocasque[1] permettant d'afficher des éléments en 3 dimensions.

Depuis cette époque, de nombreux progrès ont été faits, et la réalité augmentée est devenue un domaine de recherche à part entière. Dans les années 1990 des conférences ayant pour sujet la réalité augmentée ont commencé à apparaître : the International Workshop and Symposium on Augmented Reality, the International Symposium on Mixed Reality, the Designing Augmented Reality Environnements Workshop.

Il existe différents domaines de recherche en réalité augmentée. Tout d'abord se pose le problème du dispositif d'affichage : quelle technologie employer pour

[1] en anglais : « HMD », pour Head Mounted Display

augmenter ou diminuer les éléments présents dans un environnement réel ? Ensuite se pose la question du rendu réaliste. Il faut que l'insertion soit harmonieuse et respecte les caractéristiques de l'environnement, telles que la position des sources lumineuses, les ombres portées, les occultations entre éléments. Enfin il y a le problème de la localisation, qui consiste à déterminer précisément la position de l'utilisateur, ou du moins de la caméra, dans l'environnement de façon à modifier l'environnement au bon endroit.

1.2.1 Les dispositifs de réalité augmentée

Les dispositifs permettant d'afficher le mélange entre éléments virtuels et réels peuvent être séparés en trois catégories, en plus du simple affichage sur un écran d'ordinateur : les visiocasques, les dispositifs portables, et les projecteurs.

Les visiocasques Ce sont des dispositifs à placer sur la tête, où la visualisation se fait par l'intermédiaire d'un miroir opaque ou semi-transparent. Il en existe de deux types : ceux de vision par optique, et ceux de vision par caméra interposée. Les premiers possèdent un miroir semi-transparent qui laisse passer les informations visuelles extérieures au casque. Les éléments virtuels sont projetés sur le miroir par un dispositif intégré au casque. Les seconds possèdent un miroir opaque. Les informations extérieures sont filmées par une caméra et reprojetées, après ajout des éléments virtuels, sur le miroir (voir figure 1.2). Ces casques, disponibles dans le commerce à coût plus ou moins prohibitif, ont quelques inconvénients : leur champ de vision est assez réduit (horizontalement, de l'ordre de 30 degrés), et leur résolution assez faible (entre 180 000 et 240 000 pixels).

Les casques à vision par optique permettent de voir correctement l'environnement réel, mais avec ces dispositifs il n'est pas possible de retirer artificiellement des éléments réels. Les casques à vision par caméra le permettent, mais elles ont un défaut de parallaxe, la caméra étant légèrement décalée par rapport à la position des yeux, ce qui provoque une fatigue de l'utilisateur assez importante.

Il existe un troisième type de visiocasque, dont le dispositif consiste à projeter des images directement sur la rétine à l'aide d'un laser de faible intensité. Cette solution permet de réaliser des images ayant un fort contraste et une forte luminosité, sans nécessiter une consommation d'énergie importante [58]. Toutefois l'aspect intrusif de cette technologie peut rebuter.

Les dispositifs portables Ce type de dispositif est constitué d'un écran LCD tenu à la main, relié à une caméra. Cela fonctionne sur le même principe que le

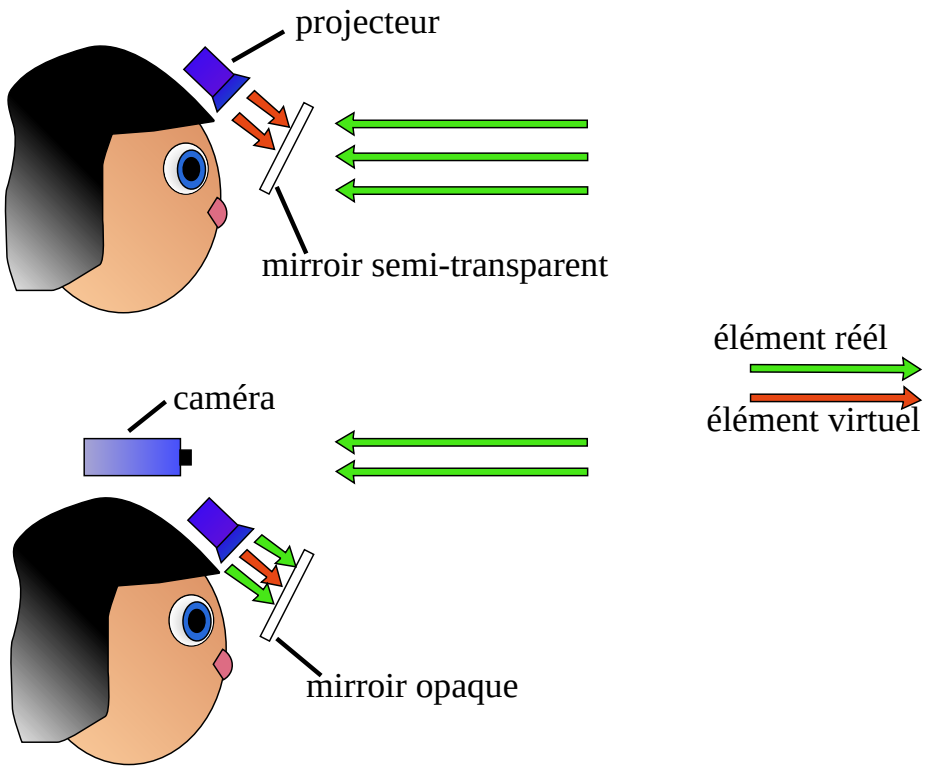


FIG. 1.2 – Visiocasques - haut : miroir transparent - bas : miroir opaque

casque avec miroir opaque : l'environnement réel est filmé par la caméra, traité pour ajouter/supprimer des éléments, puis envoyé sur l'écran. L'écran se comporte comme une fenêtre déformante du monde réel.

Les projecteurs Les deux dispositifs décrits précédemment ont l'inconvénient d'être des dispositifs personnels, non adaptés au travail collaboratif. Pour que la réalité augmentée soit visible de la même façon par plusieurs personnes, les éléments virtuels sont projetés sur l'environnement réel, et ne nécessitent pas de lunettes spéciales. Un exemple de ce type de dispositifs est le « tableau magique » de Bérard [9]. Il s'agit d'un tableau réagissant aux gestes de l'orateur, dont l'affichage est réalisé à l'aide d'un vidéo-projecteur (figure 1.3).

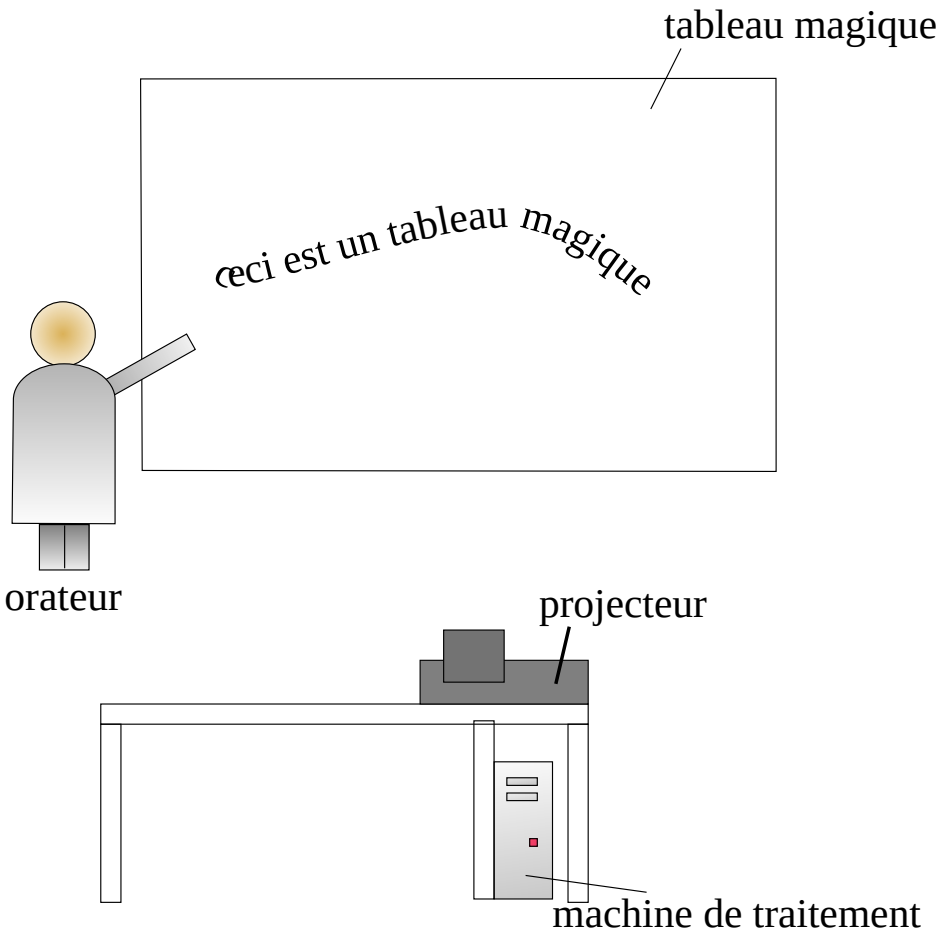


FIG. 1.3 – Le tableau magique

1.2.2 Affichage des éléments virtuels

Pour avoir un rendu réaliste, il faut que l'objet artificiel soit placé au bon endroit et qu'il soit cohérent avec son environnement, notamment en ce qui concerne le placement des sources de lumière.

Positionner correctement l'objet revient à minimiser une erreur d'estimation de pose : lorsque la pose de la caméra, ou l'estimation du champ de vision de l'utilisateur, sont mal calculés, même une très faible erreur apparaîtra de façon évidente. En effet l'œil humain est très sensible à certaines incohérences et un objet mal positionné, ou qui tremble, sera immédiatement visible.

Pour que l'insertion soit moins visible, il est par exemple possible d'utiliser

une fonction de transparence qui, par un système de fondu, va intégrer l'objet progressivement dans l'environnement réel. Si les bords de l'objet sont légèrement transparents, la transition entre éléments réels et virtuels est moins sensible [26].

Le problème de l'occultation est lui aussi complexe. Pour que le rendu soit réaliste, un objet inséré artificiellement doit occulter les objets derrière lui mais être occulté par les objets situés devant lui par rapport à la caméra. Cela ne se résume pas à établir une carte des profondeurs : il faut segmenter l'image pour retrouver les différents objets qui la compose. Lepetit [43] a ainsi développé une méthode semi-automatique d'identification d'objets et de leur position à partir de silhouettes (voir figure 1.4).

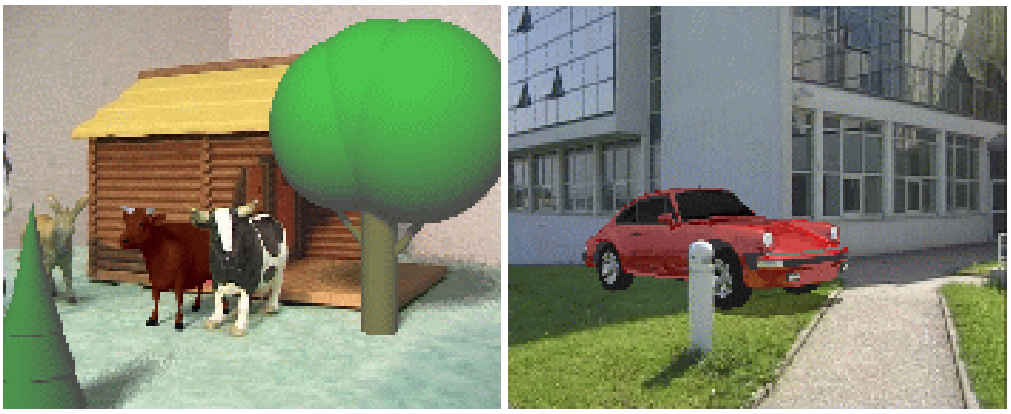


FIG. 1.4 – Gestion des occultations en réalité augmentée. La vache brune, l'arbre rond et la voiture ont été ajoutés. (images tirées du site de Vincent Lepetit)

Enfin pour que le rendu des objets virtuels soit le plus réaliste possible, il faut être capable de retrouver les caractéristiques d'illumination et de réflectance de l'environnement. Il existe de nombreux travaux sur ce domaine. Citons par exemple les travaux de Mukaigawa *et al.* [54], qui séparent et analysent les différentes sources de lumière et d'ombre d'une scène grâce à une analyse photométrique, ou ceux de Stauder [64], qui détermine les paramètres d'illumination d'une scène (voir figure 1.5).

1.2.3 Capteurs de suivi

Afin de pouvoir aligner correctement les objets virtuels et réels, il convient de déterminer très précisément le point de vue de l'utilisateur, ou la pose de la caméra.

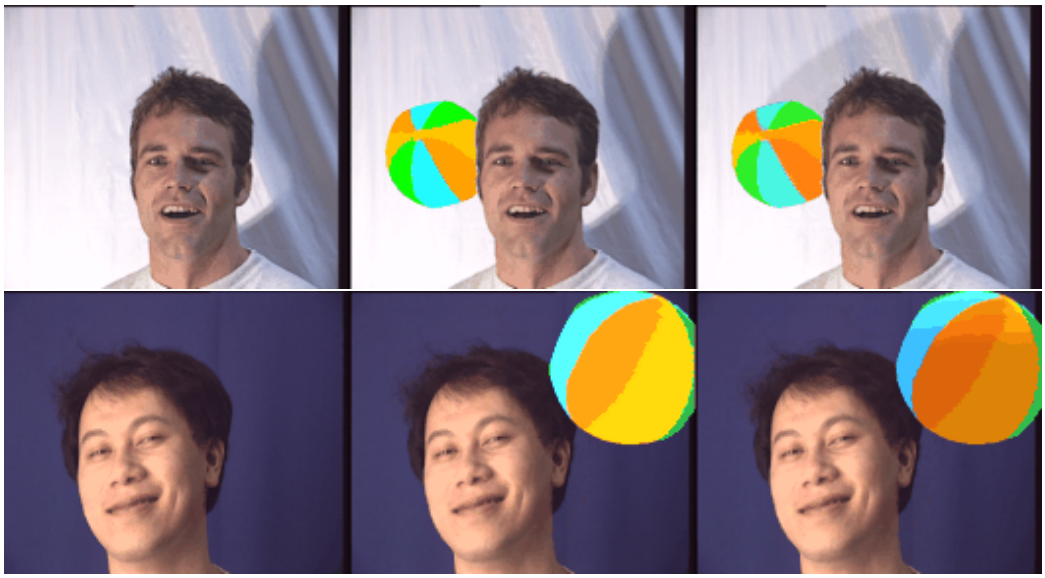


FIG. 1.5 – Détermination des paramètres d'illumination - gauche : image de départ - milieu : insertion d'une balle sans respect des paramètres d'illumination - droite : avec respect de l'illumination(d'après le site de Jürgen Stauder)

Pour connaître cette pose, cette fois encore plusieurs solutions sont possibles. La première consiste à adapter sur la caméra un système de capteurs permettant de connaître à tout instant sa position dans son environnement. Cette solution est par exemple utilisée pour le casque de réalité augmentée de Newman *et al.* [55]. Ce casque, que l'on peut voir à la figure 1.6, est muni de plusieurs dispositifs lui permettant de se localiser dans l'espace : un capteur inertiel et trois émetteurs d'ultrasons appelés « bats » (chauve-souris). Des récepteurs ultrasons disposés à des positions connues sur le plafond dans tout le bâtiment permettent de localiser les trois « bats », puis, par extension, la position du casque.

On peut constater que ce type de dispositif, qui ne fonctionne qu'en intérieur, demande des moyens matériels sophistiqués, et une installation relativement complexe. En effet c'est tout le bâtiment qui doit être équipé pour que le suivi par capteur soit efficace.

Une autre solution est le suivi par vision à l'aide de repères visuels. L'environnement est modifié par la pose d'un certain nombre de marqueurs de tailles variables disposés à des endroits connus [11], qui vont permettre de se localiser dans l'environnement. Un algorithme de traitement d'image suit ces marqueurs et calcule la pose de la caméra en temps réel.

Dans le cas de la réalité augmentée en environnement extérieur, ou si l'envi-

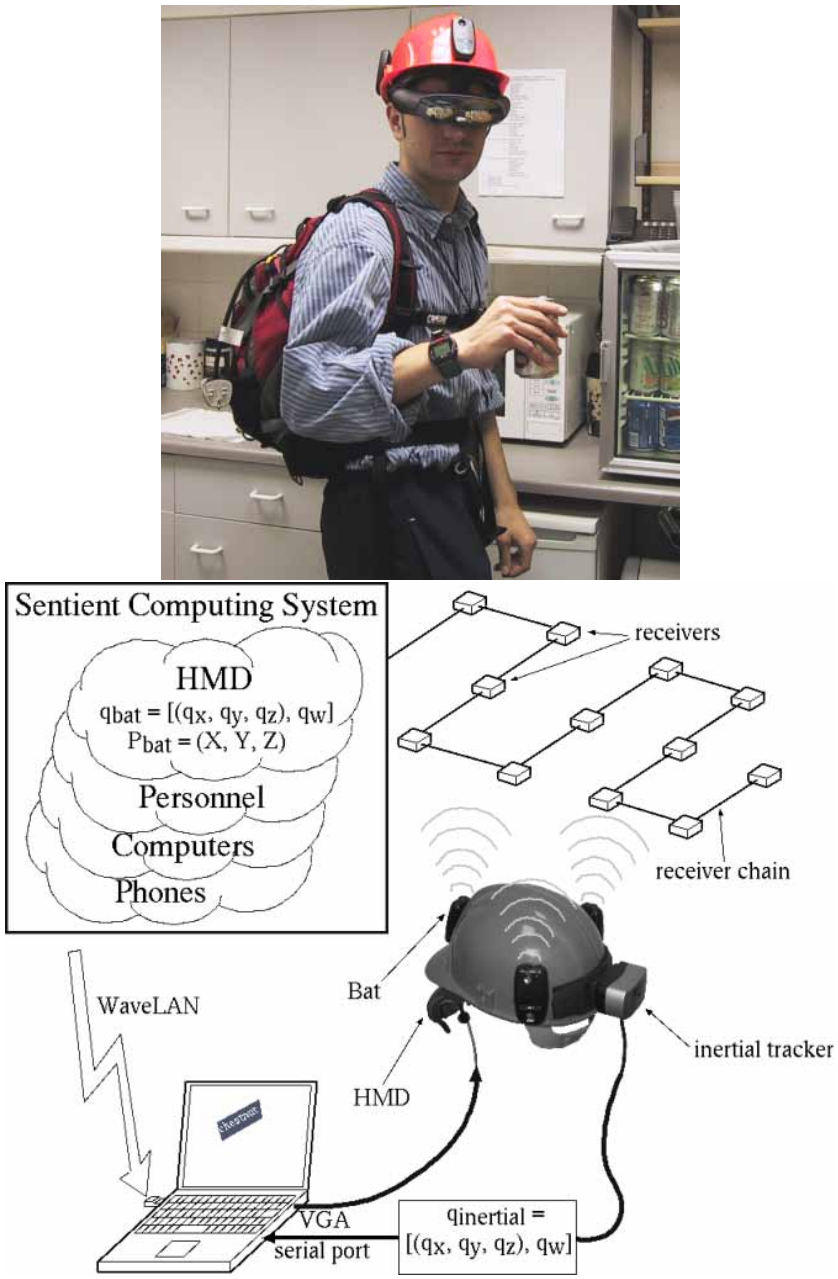


FIG. 1.6 – Dispositif de réalité augmentée de Newman *et al.* [55]

ronnement n'a pas pu être préparé en positionnant des marqueurs, il existe encore d'autres types de suivi. Il existe par exemple un système basé sur un dispositif hybride compas+gyroscope+suivi par vision permettant de se repérer en extérieur

et qui indique les noms des bâtiments actuellement dans le champ de vision de l'utilisateur et indique leur position par une flèche [5].

1.2.4 Calibration

Pour que l'ajustement des objets virtuels dans l'environnement réel soit précis, la réalité augmentée nécessite en plus une calibration rigoureuse. Cela inclut l'estimation des paramètres de caméra, du champ de vision, de la position des capteurs, des marqueurs, de la distorsion…

Il est bien sûr possible d'utiliser des méthodes de calibration par ailleurs bien connues dans d'autres domaines de la vision par ordinateur, mais des méthodes permettant de s'affranchir de cette calibration ont été développées pour la réalité augmentée, ainsi que des méthodes d'autocalibration.

Par exemple Simon *et al.* [62, 61] ont développé un système de suivi pour la réalité augmentée à base de suivi de plans, qui utilise les plans présents à l'image pour se relocaliser, et insérer des objets virtuels. Les plans, comme par exemple le plan du sol de la place dans une séquence « place Stanislas », deviennent alors des bases « naturelles » pour positionner un repère.

1.3 Le suivi temps réel

1.3.1 Introduction

Une séquence vidéo est composée d'images fixes défilant suffisamment vite pour que l'œil humain ne perçoive pas une succession d'images mais ait une illusion de mouvement (figure 1.7). Pour un bon confort visuel, 24 images par secondes est la norme choisie pour le cinéma. On trouve dans le commerce des caméras fonctionnant à 30 images par seconde, et certaines caméras spécifiques développées en laboratoire peuvent même atteindre 1000 images par seconde [10].

Deux images successives d'une séquence vidéo seront très similaires, et cela d'autant plus si la fréquence de la caméra ayant généré cette séquence est élevée. Lors du traitement d'une séquence vidéo, un élément dont on connaît la position dans une image de la séquence sera pratiquement à la même position à l'image précédente et à l'image suivante.

Ainsi il peut être judicieux, si l'on veut connaître la position exacte d'un élément, de se baser sur cette hypothèse pour estimer cette position. Plutôt que de parcourir toute l'image à la recherche de cet élément, on utilise les informations issues de traitements précédents et l'hypothèse selon laquelle il y a eu peu de

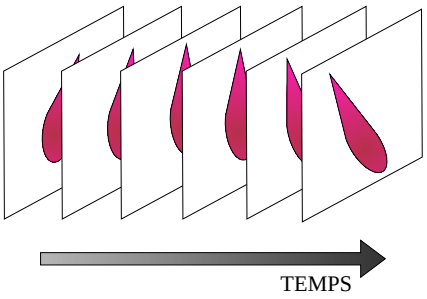


FIG. 1.7 – Décomposition d'une séquence vidéo

changement depuis l'image précédente.

Une autre façon de procéder est d'utiliser un modèle du mouvement : par exemple si l'on sait que l'élément effectuait une translation de la gauche vers la droite pour les images 1 à n de la séquence, on peut raisonnablement estimer qu'il a poursuivi sa trajectoire au moment où a été prise l'image $n+1$, et ainsi avoir une estimation de sa nouvelle position dans cette image.

Les techniques de suivi sont tout de même liées aux techniques de détections, car elles ont besoin d'être initialisées avec la position de départ de l'élément à suivre.

Elles sont aussi liées aux techniques d'asservissement, notamment d'asservissement visuel. L'asservissement consiste à contrôler les mouvements d'un système dynamique grâce à des informations fournies par des capteurs, ces capteurs étant des caméras dans le cas de l'asservissement visuel. D'une certaine façon, les algorithmes d'asservissement sont le dual des algorithmes de suivi : là où les algorithmes de suivi cherchent à déterminer les mouvements effectués par l'objet-cible, les algorithmes d'asservissement cherchent à déplacer l'objet pour que sa nouvelle position corresponde à une position désirée. Ainsi les problématiques du suivi et de l'asservissement sont souvent proches, et l'on retrouve dans les travaux d'asservissement robotique des techniques et suivi, et réciproquement, comme par exemple dans les travaux d'Éric Marchand [50].

1.3.2 Principes généraux

Les algorithmes de suivi doivent répondre à un certain nombre de critères pour pouvoir être estimés satisfaisants :

– Ils doivent être capables de suivre des mouvements conséquents de l'objet,

avec des vitesses et des accélérations variables ;
- Ils ne doivent pas être perturbés par les éléments extérieurs à l'objet suivi ;
- Ils doivent être stables, c'est-à-dire capables de suivre sur une durée suffisamment étendue sans perdre l'objet ;
- Ils doivent être robustes à de légères occultations ;
- Ils doivent avoir un temps d'exécution compatible avec les contraintes pour lesquels ils sont utilisés (par exemple être compatibles avec les fréquences élevées nécessaires en robotique) ;
- Ils doivent renvoyer une localisation suffisamment précise pour être adaptée aux applications visées.

D'une manière générale, les algorithmes de suivi d'objets dans des images vidéo peuvent être vus comme des algorithmes d'optimisation. Il s'agit de déterminer les paramètres d'un vecteur d'état de manière à optimiser le *recalage* du modèle sur une image. Ces paramètres d'état peuvent comporter des informations sur la position, l'orientation, les vitesses de l'objet à suivre, et peuvent éventuellement décrire des changements d'aspect ou de propriété des objets.

Ces algorithmes peuvent être représentés par une boucle à trois étapes (figure 1.8) :

1. Prédiction de la position de l'objet, connaissant certaines informations sur sa position précédente et son mouvement ;
2. Mise en correspondance, dans la zone prédite, des primitives de l'objet afin de trouver sa position précise ;
3. Mise à jour des paramètres ;
4. Retour à l'étape 1.

Comme on l'a vu précédemment, on considère que la cible s'est peu déplacée, ou alors selon un déplacement cohérent avec un modèle dont les paramètres sont estimés à partir des images précédentes. Ainsi les variations des paramètres d'état seront faibles d'une image à l'autre, et prévisibles connaissant les paramètres de déplacement du modèle.

Les informations utilisées pour le suivi peuvent être de deux types : à la fois les primitives de l'image, c'est-à-dire les informations locales de couleur, forme, texture, mais aussi les informations liées au déplacement de la cible : vitesse, accélération.

Il existe de nombreux algorithmes permettant de suivre des objets dans des séquences vidéo, et c'est un domaine de recherche encore ouvert. Obtenir une méthode de suivi à la fois robuste, précise et efficace est difficile.

Ces algorithmes ont deux composantes distinctes : la phase de prédiction connaissant les paramètres de mouvement de l'objet, et la phase de mise en cor-

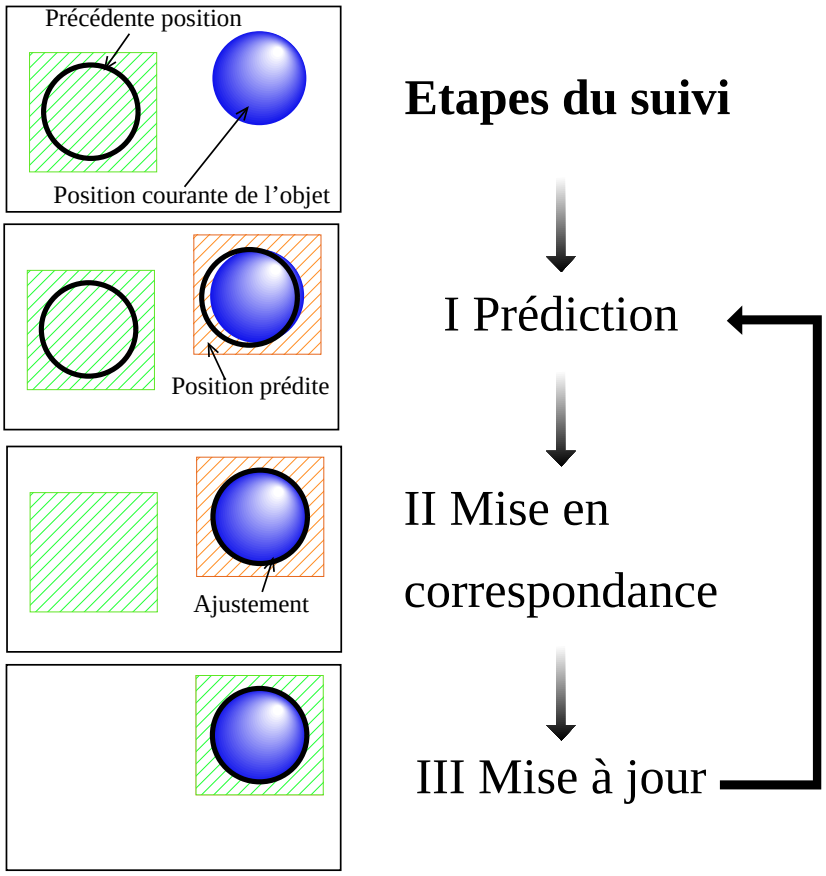


FIG. 1.8 – suivi : principe général

respondance d'informations locales. La plupart des algorithmes utilisent ces deux composantes, même si certains privilégient l'une par rapport à l'autre.

Ainsi nous étudions ces deux composantes dans deux sections différentes, en exposant dans chacune les algorithmes représentatifs de ce qui se fait dans ce domaine.

La première contient ceux qui se basent particulièrement sur la prédiction de la position suivante de l'objet connaissant ses paramètres de mouvement, tels que le filtre de Kalman [39, 32], ou les filtres à particules [3, 25] (Étape 1 de la boucle).

La seconde contient ceux qui sont plus axés sur la mise en correspondance d'éléments de l'objet permettant retrouver sa position, ces éléments pouvant être des mesures servant à localiser un modèle 2D ou 3D connu de la cible [47, 23], ou des primitives telles que la couleur ou les contours [13, 33].

1.3.3 Détection d'objets

La détection de l'objet-cible est souvent la phase initiale du suivi. Elle est plus coûteuse en temps de calcul que le suivi lui-même, mais ne demande pas de connaissance *a priori* sur la position de la cible. Avec les progrès technologiques réalisés ces dernières années en informatique et grâce à des algorithmes de classification efficaces, il est désormais possible d'exécuter des algorithmes de détection fonctionnant en temps réel vidéo [44].

Il existe différentes différentes sortes de détection.

Une première technique est la détection de forme. L'image reçoit tout d'abord un pré-traitement permettant supprimer le bruit, de faire ressortir les contours ou de la segmenter en éléments pertinents puis de les rendre invariants aux rotations, translation et changement d'échelle, ou toute combinaison des traitement précédemment cités. Le résultat de ce pré-traitement est ensuite comparé au contenu d'une base de données. Cette technique permet de retrouver des images ayant des caractéristiques en commun [24].

Si un modèle de l'objet est disponible, la détection consiste à mettre en correspondance les primitives du modèle avec des primitives détectées dans l'image. Selon que le modèle est fixe ou déformable, cette détection peut se faire de différentes façons.

Si le modèle est fixe, c'est-à-dire s'il ne varie pas quelle que soit la pose de la caméra, il existe deux techniques pour réaliser la détection. La première consiste à considérer différentes positions de l'objet dans l'image à traiter jusqu'à trouver celle qui minimise une distance avec le modèle connu. Cette technique, peu coûteuse en temps de calcul, est bien adaptée à des images prises dans les mêmes conditions de luminosité, avec peu de bruit, et des angles de vue identiques. La seconde est la corrélation. Elle consiste à calculer les facteurs de corrélation croisée (de l'image cible vers l'image où la position de l'objet est connue, puis de cette image vers l'image cible) normalisés (pour l'invariance aux changements d'intensité) pour toutes les positions possibles de l'objet. L'objet recherché se situe à la position donnant le meilleur critère de corrélation. Cette méthode est très coûteuse en temps de calcul mais plus robuste au bruit et aux changements d'illumination que la précédente.

Dans le cas où le modèle est déformable, que ce soit par des déformations rigides, comme des changements de pose de la caméra, ou non rigides, la détection est nettement moins aisée. Dans ce cas, en plus de la corrélation, il faut tester les déformations possibles du modèle, ou ses variations d'aspect. Évidemment tester toutes les déformations possibles sur toutes les positions possibles est extrêmement coûteux. Toutefois, en connaissant *a priori* un modèle statistique des

déformations de l'objet-cible, et en procédant itérativement de façon à trouver progressivement les bons paramètres de déformations, il est possible de détecter des objets déformables [31].

1.3.4 Estimation des paramètres de mouvement

Il s'agit de techniques basées sur le filtrage, appliquées à la vision par ordinateur. Ces techniques ne sont pas spécifiques à la vision et ont même parfois été développées pour de toutes autres applications. Le filtre de Kalman [39], par exemple, est un outil de filtrage utilisé aussi bien pour le suivi radar, la commande et même la finance [59].

Les méthodes de suivi probabilistes sont fréquemment employées dans le domaine de la vision par ordinateur. Les techniques de filtrage particulaire, notamment, ont été beaucoup utilisées ces dernières années.

Elles consistent à prédire les paramètres d'un modèle (la position d'un objet, ou une zone d'incertitude concernant la position de cet objet), en utilisant des mesures faites dans des images précédentes.

Le problème du suivi est posé de la façon suivante :

Les paramètres du modèle sont représentés sous la forme d'un vecteur d'état, ici noté x_k :

$$x_k = f_k(x_{k-1}, v_{k-1})$$

autrement dit, le vecteur x à l'instant k est fonction de son état à l'instant $k-1$ et d'un bruit v_{k-1}.

L'objectif du suivi est alors d'estimer récursivement x_k à partir de mesures

$$z_k = h_k(x_k, n_k)$$

où z_k, la mesure à l'instant k, est fonction de x_k et d'un bruit n_k.

Dans le cadre de la vision par ordinateur, cette mesure pourra être une distance dans l'image, des différences d'intensité lumineuse...

Dans la perspective bayésienne, le problème du suivi se ramène alors à un calcul récursif du degré de confiance dans le vecteur d'état x à l'instant k, connaissant chacun des z précédents ($z_{1:k}$).

Il faut alors construire récursivement la densité de probabilité $p(x_k \mid z1 : k)$, sachant que $p(x_0 \mid z_0) \equiv p(x_0)$ est supposée connue. Cela se fait en deux étapes : prédiction et mise à jour.

La prédiction est réalisée en utilisant la densité de probabilité au temps $k-1$ ($p(x_{k-1} \mid z_{1:k-1})$). La mesure à l'instant k, z_k, permet alors la mise à jour :

$$p(x_k \mid z_{1:k}) = \frac{p(z_k \mid x_k)p(x_k \mid z_{1:k-1})}{p(z_k \mid z_{1:k-1})}$$

L'algorithme optimal pour réaliser ces opérations est le filtre de Kalman. Mais l'application du filtre de Kalman nécessite que certaines hypothèses fortes soient validées, ce qui n'est pas toujours possible (densités de probabilité gaussiennes à chaque étape du raisonnement, fonctions f_k et h_k linéaires). C'est pour cela qu'un algorithme sous-optimal est le plus souvent employé, le filtre de Kalman étendu, qui approxime les densités de probabilité par des gaussiennes et considère que f et h, quoique non linéaires, peuvent être linéarisées par un développement de Taylor à l'ordre 1.

Il existe des alternatives au filtre de Kalman étendu, et l'on a vu se développer ces dernières années des algorithmes de suivi basés sur des méthodes de type Monte-Carlo. Celles-ci peuvent se présenter sous plusieurs noms : filtres à particules [56], algorithme de condensation [30], ...

L'idée principale, derrière ces méthodes, est de représenter la fonction de densité de probabilité par un jeu d'échantillons sélectionnés aléatoirement et pondérés. Les estimations sont alors réalisées à partir de ces échantillons et de ces poids. Plus le nombre d'échantillons est élevé, et plus l'on se rapproche de la représentation continue usuelle.

Ces méthodes sont plus souples que le Kalman étendu, puisqu'il est possible de jouer sur le nombre d'échantillons pour augmenter la précision du suivi ou au contraire diminuer le temps de calcul.

Il est aussi possible de représenter, à l'aide de ces particules, plusieurs densités de probabilité. On pourra alors suivre plusieurs éléments simultanément. C'est par exemple ce que font Zhao et Nevatia [75] pour le suivi des piétons au milieu d'une foule.

1.3.5 Suivi par association de données

Dans la section précédente nous avons vu comment utiliser des densités de probabilité pour estimer itérativement des paramètres d'un modèle de mouvement, connaissant la valeur de ces paramètres aux itérations précédentes. Nous avons parlé de « mesure » de façon générique, il est temps maintenant de parler des mesures spécifiques à la vision.

Suivant le type de suivi à réaliser, ces mesures peuvent porter sur différents types de données, et être associées à différentes informations de l'image. Nous allons voir dans cette section différentes informations de l'image pouvant être utilisées.

Suivi par fenêtre de corrélation

C'est le suivi le plus simple. Il consiste à apparier, d'une image à l'autre, de petites « fenêtres » de l'image suivant un critère de type corrélation.

Connaissant la zone dans laquelle doit se situer la fenêtre recherchée dans l'image courante, toutes les fenêtres possibles sont testées et celle qui est la plus corrélée à la fenêtre de départ est considérée comme étant la nouvelle position de la zone suivie.

Suivant les transformations possibles des fenêtres initiales, les distances utilisées devront prendre en compte des translations, des rotations, des changements d'échelle, des transformations affines, des transformations homographiques... Plus la transformation envisagée est complexe, et plus le nombre de fenêtres candidates augmentent, et du même coup le temps de calcul.

De plus ce type de suivi de donne pas d'information sur le mouvement global d'un objet, seulement des déplacements de zones de l'image.

Ces fenêtres sont l'une des primitives, c'est-à-dire un élément géométrique simple, qui peuvent être utilisées en vision. Il en existe d'autres que nous allons voir dans la section suivante.

Suivi à partir de primitives

Les primitives peuvent être de différents types : des points [20], des contours [17], des couleurs, des niveaux de gris,...

L'utilisation de ces primitives peut être une première étape pour recaler un modèle, comme on le verra dans la section suivante, mais peut aussi permettre directement d'effectuer des tâches de suivi.

L'algorithme de Lucas et Kanade [49] sur le suivi à l'aide de primitives, publié en 1981, est l'un des algorithmes fondamentaux de ce domaine. Il a été suivi d'articles développant la méthode à utiliser ainsi que la façon de trouver de « bonnes » primitives [67, 60]. Les primitives décrites dans ces articles sont des points d'intérêt, mais il existe des algorithmes utilisant d'autres types de primitives.

Ainsi, Comaniciu *et al.* [13] réalisent du suivi en temps réel d'objets non-rigides en utilisant comme information les couleurs des pixels de la zone de l'image où se trouve l'objet suivi.

Il s'agit donc cette fois encore d'éléments simples : la couleurs des pixels sur une région donnée. Afin de pouvoir utiliser ces primitives pour suivre un objet, elles sont traitées de façon à en obtenir des informations fiables.

Les informations de couleurs sont pondérées par la distance des pixels par rapport au centre de la zone suivie, afin que les pixels éloignés du centre, les plus

susceptibles d'être occultés ou déformés, soient moins pris en compte que les pixels proches du centre. Ces couleurs pondérées sont alors utilisées pour créer un histogramme représentant la distribution des couleurs sur la zone à suivre.

Une distance, dite distance de Bhattacharyya, est ensuite calculée en utilisant cet histogramme, afin de déterminer la dissimilarité entre la zone à suivre et la zone courante. Par une méthode itérative probabiliste baptisée « Mean Shift », l'algorithme trouve la zone la plus similaire à la zone recherchée après quelques itérations, c'est-à-dire la zone pour laquelle la distance de Bhattacharyya est la plus faible (figure 1.9).

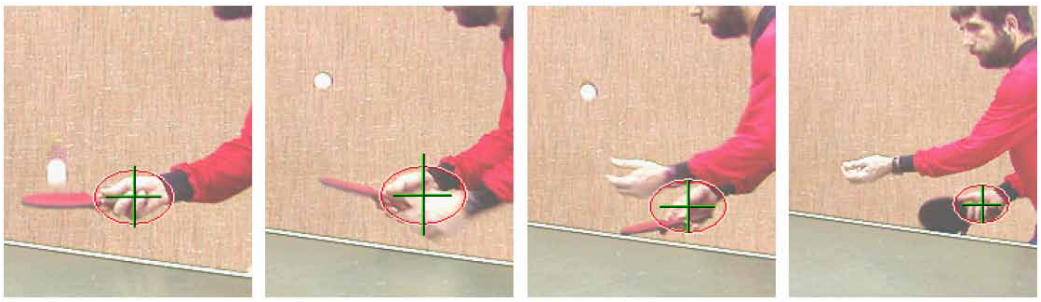


FIG. 1.9 – Suivi d'une main par méthode de Comaniciu *et al.* [13]

Jepson *et al.* [33] proposent une approche qui permet de mettre à jour le motif suivi. Le modèle du motif est obtenu par application d'une version *en ligne* de l'algorithme EM (*Espérance Maximisation*). Cet algorithme itératif est utilisé pour modéliser les réponses de filtres orientables multi-échelles, appliqués en chaque point du motif.

Cette fois les primitives utilisées sont des ondelettes, et l'algorithme est basé sur des « pyramides orientables », c'est-à-dire des régions de l'image vues suivant plusieurs niveaux d'échelle, ce qui permet à l'algorithme d'être robuste aux variations de taille de l'objet (figure 1.10).

Il existe d'autres primitives qui peuvent être utilisées, comme les segments. Dans leur article de 1990, Deriche et Faugeras [21] exposaient ainsi une méthode de suivi de segments basée sur le filtre de Kalman. Le modèle cinématique utilisait la trajectoire, la vitesse et l'accélération de quatre points.

L'ensemble des travaux basés sur du *template matching* rentrent aussi dans cette catégorie. Nous pouvons par exemple mentionner les travaux de La Cascia *et al.* [41] sur le suivi de visage. La texture du visage est plaquée sur un cylindre 3D et la pose de ce cylindre est optimisée de manière à ce que la différence entre la texture du visage dans l'image et la texture de la projection du

FIG. 1.10 – Suivi d'un visage par méthode de Jepson *et al.* [33]

cylindre ait une norme Euclidienne la plus faible possible.

Dans ces méthodes il manque au résultat obtenu des informations sur l'attitude de l'objet suivi. Ainsi on connaît certes la position de la main du joueur de ping-pong, mais pas les paramètres de position de cette main : les doigts sont-ils ouverts ou fermés ? La main est-elle vue de profil, de trois-quarts ?

Pour obtenir ces informations supplémentaires, il faut alors connaître des informations sur l'objet suivi qui permettent d'estimer cette attitude, comme un modèle de l'objet.

Suivi à partir d'un modèle connu

Il s'agit de suivre un objet dont le modèle, principalement un modèle CAO, est parfaitement connu. Bien entendu des primitives locales sont là aussi employées, mais cette fois comme moyen de recaler au mieux le modèle.

Lowe avait décrit les bases du suivi robuste d'un objet dont le modèle est connu dès 1992 [47], dans un article dans lequel il expliquait une méthode pour prendre en compte les erreurs dans les appariements de primitives utilisés par le suivi.

FIG. 1.11 – Suivi d'un livre par méthode de Lowe [47]

Depuis, d'autres méthodes de suivi utilisant un modèle ont été développées. Par exemple Drummond et Cipolla utilisent un modèle CAO de l'objet suivi [23]. Celui-ci est représenté par l'intermédiaire d'un arbre où chaque nœud correspond à un plan de l'objet, et contient une liste des arêtes et des polygones convexes de ce plan. L'algorithme utilisé pour le suivi est basé sur le suivi d'arêtes, sur un algorithme de type *snake* (*contrained snake*[12]).

D'autres travaux sont basés sur le suivi temps réel d'objets dont le modèle CAO est connu, comme par exemple ceux de Comport *et al.* [17, 16], ou ceux de Wunsch et Hirzinger [73], et appliqués à l'asservissement de robots.

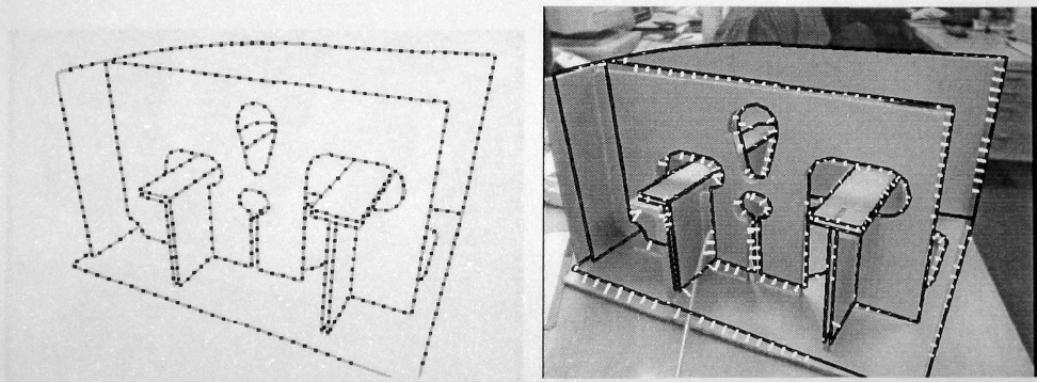


FIG. 1.12 – Suivi d'un objet de modèle CAO connu par Drummond et Cipolla [23]

D'autres travaux, ceux de Kollnig et Nagel [40] portent sur le suivi de véhicules à partir de modèles polyédriques 3D des véhicules. L'originalité de ces travaux est d'utiliser l'information du gradient plutôt que de détecter les contours. La pose 3D du modèle est optimisée de manière à maximiser la probabilité qu'un point du modèle se projette sur un point de contour (gradient élevé) de l'image.

On peut aussi citer les travaux de Marchand *et al.* [51] qui utilisent les contrastes d'intensité dans des images pour recaler le modèle CAO de l'objet suivi, et ceux de Yao et Cham [74] qui utilisent un modèle générique de tête, déformé afin de correspondre à la tête observée, pour suivre des visages sur des séquences de film.

Mais ces méthodes ne sont pas les seules à permettre d'obtenir un résultat précis. La méthodes de suivi de texture décrite dans la section suivante en est un exemple.

1.4 Suivi planaire de motifs texturés

Hager *et al.* [27] ont développé une approche de suivi de texture permettant de suivre des motifs texturés en temps réel. Limitée à des objets planaires et à de petits déplacements, elle a été étendue par Jurie et Dhome au cas de mouvements importants [37] et aux cas de surfaces gauches [35].

Benhimane et Malis ont étendu cette méthode au calcul d'une minimisation du second ordre [7], et l'ont ensuite utilisée pour la commande de véhicules intelligents [8].

Il s'agit d'un suivi de plan texturé temps réel fonctionnant en deux temps, une phase d'apprentissage hors-ligne et une phase de suivi temps réel. La texture du motif suivi est apprise sur une image contenant ce motif, puis le suivi se base sur l'hypothèse qu'il existe une transformation linéaire qui, au voisinage de cette position initiale, relie les paramètres de position du motif et les différences d'intensité lumineuse sur les pixels du motif.

Dans le domaine du suivi par vision, on considère que l'objet suivi a subi de faibles modifications entre deux images consécutives. Ainsi, par un changement de repère adéquat, il est possible de considérer le déplacement entre chaque image d'une séquence comme un déplacement au voisinage de la position initiale.

Ainsi l'hypothèse de base peut être étendue sur une longue séquence vidéo où le motif se déplace.

Le motif suivi est considéré comme un rectangle, défini par ses quatre coins, et dont la position sur une image qui servira de référence est connue précisément.

On dit d'un motif qu'il est « texturé » lorsqu'il comporte des gradients d'intensité lumineuse permettant d'obtenir des informations utiles. Le motif de la figure 1.13, par exemple, est un motif texturé, tandis qu'un texte noir sur fond blanc n'est pas un motif texturé.

1.4.1 Algorithme

Nous allons décrire ici les deux phases de cet algorithme : la phase d'apprentissage et la phase de suivi.

Phase d'apprentissage : Elle consiste à apprendre la fonction linéaire qui lie le déplacement à un changement de primitive locale. Plutôt que de chercher à calculer cette fonction, elle est apprise en générant pseudo-aléatoirement un ensemble de déplacements du motif en mesurant les variations locales correspondantes, et résolvant par une minimisation au sens des moindres carrés le système correspondant.

La texture du plan suivi est décrite par n points $(p_1, p_2, \ldots, p_n)$, où $p_i = (x, y)$. Ils forment le vecteur $\mathbf{P}$. Les p_i points sont sélectionnés de façon à respecter 2 contraintes : être répartis sur toute la surface du plan, et être positionnés de préférence là où le gradient de l'image est élevé.

En pratique, le plan est découpé en baquets et un nombre donné de points est choisi dans chaque baquet. Cette sélection peut par exemple se faire en tirant pseudo-aléatoirement des points dans chaque baquet, et en ne conservant que ceux ayant le gradient le plus élevé. L'intensité lumineuse en chacun de ces points est stockée dans le vecteur $\mathbf{V}(0)$.

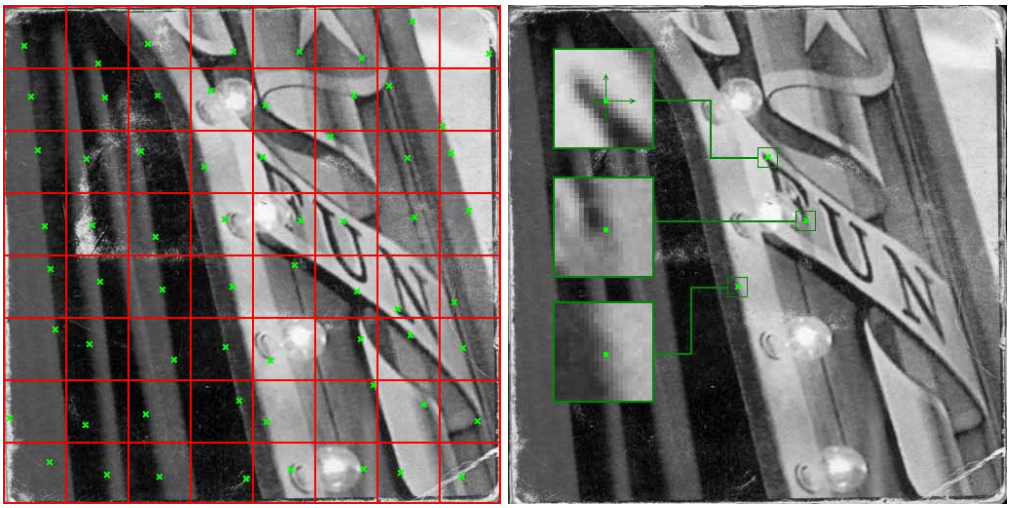


FIG. 1.13 – Sélection de points - gauche : sélection d'un point par baquet - droite : voisinage des points sélectionnés

Une telle sélection de points peut être observée à la figure 1.13. On peut y voir que l'intensité lumineuse en chaque point, dans une image texturée, peut donner une information de position : suivant si la différence d'intensité lumineuse, entre un point à sa position initiale et ce point après déplacement, est positive, négative, de valeur absolue faible ou élevée, on peut obtenir un indice sur le déplacement effectué. Ainsi sur la figure le point du bas donne une information d'abscisse : si le nouveau point est plus clair, c'est que le motif s'est déplacé vers la gauche, si il est plus foncé, le déplacement s'est effectué vers la droite. Et chaque point donne une information différente.

De façon intuitive, on comprend que la somme des informations données par les points de contrôle va permettre d'obtenir une estimation du déplacement effectué par le plan.

Soit $\mu(t) = (\mu_1(t), \mu_2(t), \ldots, \mu_p(t))$ le vecteur de paramètres décrivant la position du plan à l'instant t dans le référentiel du patch. Ces paramètres correspondent aux paramètres d'une fonction f qui amène les points de la position courante à la position estimée. μ peut décrire toute sorte de transformations planaires, mais la façon la plus efficace de procéder est de lui faire décrire une homographie, qui est la transformation la plus exacte pour expliquer des déplacements d'un plan dans l'espace projectif.

On applique m « petites perturbations » à la position initiale du patch. Cela revient à générer m transformations homographiques du plan. Pour travailler avec

des paramètres homogènes, l'homographie n'est pas décrite comme les 8 cœfficients d'une matrice 3x3, mais comme les translations des quatre coins décrivant la position du motif suivi. Ainsi tous les paramètres de la transformation sont du même ordre de grandeur.

exemple : l'homographie

$$\mathbf{Hom} = \begin{pmatrix} 0.812812 & 1.11003 & -2.29185 \\ 0.95206 & 2.0828 & -0.227538 \\ -1.88349 & -0.243421 & 1 \end{pmatrix}$$

est la transformation qui amène les quatre coins du carré $[(0,0),(0,1),(1,1),(1,0)]$ à la position $[(-2.29,-0.23),(1.67,-0.82),(0.33,-2.49),(-1.56,2.45)]$:

$$\begin{pmatrix} -2.29 & 1.67 & 0.33 & -1.56 \\ -0.23 & -0.82 & -2.49 & 2.45 \\ 1 & 1 & 1 & 1 \end{pmatrix} = \mathbf{Hom}. \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Ainsi plutôt que de décrire l'homographie par des cœfficients de matrice, elle sera décrite par les nouvelles positions des coins du rectangle $[(0,0),(0,1),(1,1),(1,0)]$, et la matrice **Hom** précédente sera décrite par le vecteur :

$$\mu_{\mathbf{Hom}} = \begin{pmatrix} -2.29 \\ -0.23 \\ 1.67 \\ -0.82 \\ 0.33 \\ -2.49 \\ -1.56 \\ 2.45 \end{pmatrix}$$

L'apprentissage du motif se fait alors de la façon suivante : Soit $\delta\mathbf{V}(i) = \mathbf{V}(0) - \mathbf{V}(i)$ le vecteur des différences d'intensité, calculé en faisant la différence entre le vecteur des intensités à la position initiale et le vecteur à la nouvelle position, après la $i^{\text{ième}}$ perturbation. Soit $\delta\mu(i)$ le vecteur décrivant la $i^{\text{ième}}$ perturbation.

Puisque nous considérons que la relation entre le déplacement et les valeurs d'intensité est linéaire[2], cela signifie qu'il existe une matrice constante $\mathbf{H}$ qui vérifie :

$$\delta\mu(i) = \mathbf{H} \,.\, \delta\mathbf{V}(i) \tag{1.1}$$

[2] à condition d'être au voisinage de la position initiale, bien sûr

La phase d'apprentissage consiste donc à déterminer la matrice de régression **H** en résolvant le système suivant :

$$\begin{pmatrix} \delta\mu_0(0) & \dots & \delta\mu_m(0) \\ \vdots & & \vdots \\ \delta\mu_0(p) & \dots & \delta\mu_m(p) \end{pmatrix} = \mathbf{H}.\begin{pmatrix} \delta\mathbf{V}_0(0) & \dots & \delta\mathbf{V}_m(0) \\ \vdots & & \vdots \\ \delta\mathbf{V}_0(n) & \dots & \delta\mathbf{V}_m(n) \end{pmatrix}$$

ou, en notation matricielle :

$$\mathbf{M}_\mu = \mathbf{H}.\mathbf{M}_\mathbf{V}$$

Cette résolution se fait en calculant la de la matrice des vecteurs $\delta\mathbf{V}$, $\mathbf{M}_\mathbf{V}$. Pour réaliser cette opération, une décomposition en valeurs singulières est réalisée :

$$\mathbf{M}_\mathbf{V} = \mathbf{U}_{\mathbf{m}\times\mathbf{m}}.\mathbf{D}_{\mathbf{m}\times\mathbf{n}}.\mathbf{V}^\mathbf{t}_{\mathbf{n}\times\mathbf{n}}$$

où $\mathbf{U}_{\mathbf{m}\times\mathbf{m}}$ et $\mathbf{V}_{\mathbf{n}\times\mathbf{n}}$ sont des matrices orthogonales, et $\mathbf{D}_{\mathbf{m}\times\mathbf{n}}$ est une matrice diagonale.

Cette décomposition permet d'obtenir facilement la pseudo-inverse de Moore-Penrose de la matrice et donc de résoudre le système :

$$\mathbf{M}_\mathbf{V}{}^\dagger = \mathbf{V}_{\mathbf{n}\times\mathbf{n}}.\mathbf{D}^\dagger_{\mathbf{m}\times\mathbf{n}}.\mathbf{U}^\mathbf{t}_{\mathbf{m}\times\mathbf{m}}$$

$$\mathbf{H} = \mathbf{M}_\mu.\mathbf{V}_{\mathbf{n}\times\mathbf{n}}.\mathbf{D}^\dagger_{\mathbf{m}\times\mathbf{n}}.\mathbf{U}^\mathbf{t}_{\mathbf{m}\times\mathbf{m}}$$

La décomposition en valeurs singulières étant une opération coûteuse, la résolution du système linéaire doit se faire hors-ligne.

Une seule matrice de régression **H** n'est toutefois pas suffisante pour réaliser un suivi efficace. Plusieurs matrices $\mathbf{H_i}$ sont utilisées, elles correspondent à des déplacements plus ou moins importants. Cela revient à se placer dans une optique multi-échelle, chaque matrice représentant l'apprentissage des déplacements du motif à une certaine échelle.

Un diagramme récapitulatif de la phase d'apprentissage est visible à la figure 1.14.

Phase de suivi : La phase de suivi utilise ensuite la matrice **H** apprise à la phase précédente. Comme nous allons le voir, elle est très simple : Nous calculons le vecteur de différence d'intensité $\delta\mathbf{V}$ comme expliqué précédemment. Ce vecteur permet d'obtenir $\delta\mu$, une estimation des paramètres de déplacement dans le référentiel initial, en utilisant l'équation (1.1).

Cette équation n'étant valable qu'au voisinage de la position initiale du patch, on change de référentiel pour se placer dans le référentiel du motif, en utilisant :

$$\mathbf{F}(\mu(t)) = \mathbf{F}(\mu(t-1)) \circ \mathbf{F}(\delta\mu(t)),$$

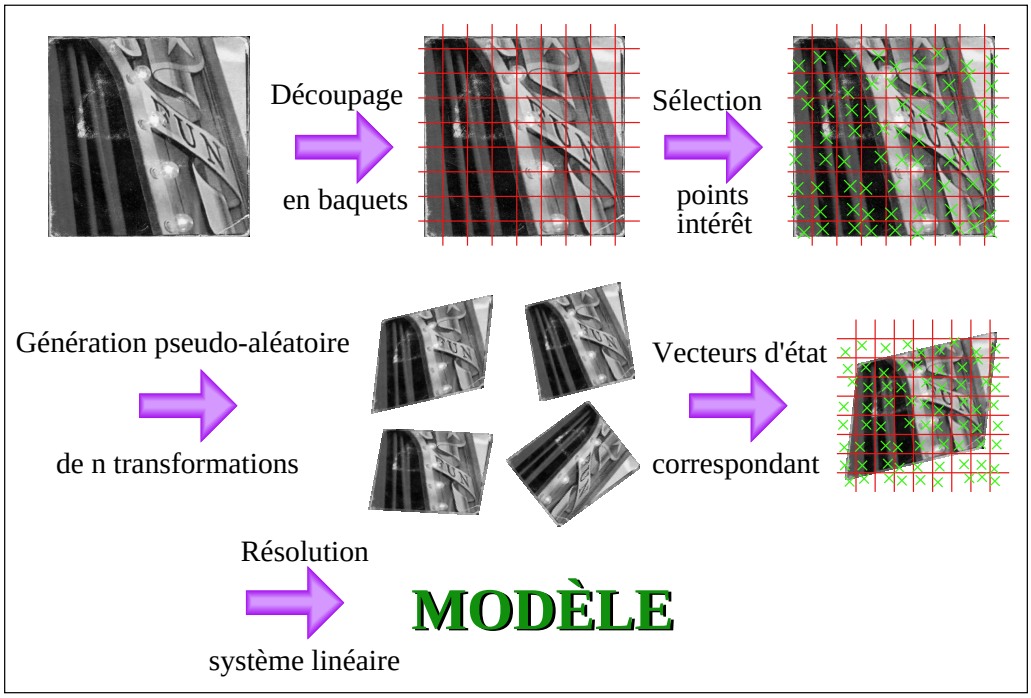


FIG. 1.14 – Phase d'apprentissage pour le suivi plan.

$\mathbf{F}(\mu)$ étant la matrice 3×3 obtenue à partir du vecteur de paramètres de transformation μ.

Comme il n'y a pas une, mais plusieurs matrices $\mathbf{H_i}$, cette opération est effectuée autant de fois qu'il y a d'échelles, en commençant par l'échelle la moins précise pour aller jusqu'à la plus précise.

On peut vite s'apercevoir que la complexité algorithmique de cette approche est très faible : uniquement une multiplication matricielle et plusieurs opérations mathématiques simples (addition, *etc.*). C'est pour cette raison que cet algorithme est très efficace.

En plus de son efficacité, cet algorithme a aussi l'avantage d'être précis ; la régression donnée en éq. 1.1 permet d'obtenir une précision allant jusqu'au subpixellique dans le calcul des paramètres de déplacement.

1.4.2 Une application : jeu de labyrinthe en réalité augmentée

Afin de mieux appréhender cet algorithme, j'ai réalisé au début de ma thèse une petite application de réalité augmentée qui l'utilise. Il s'agit d'un jeu basé sur

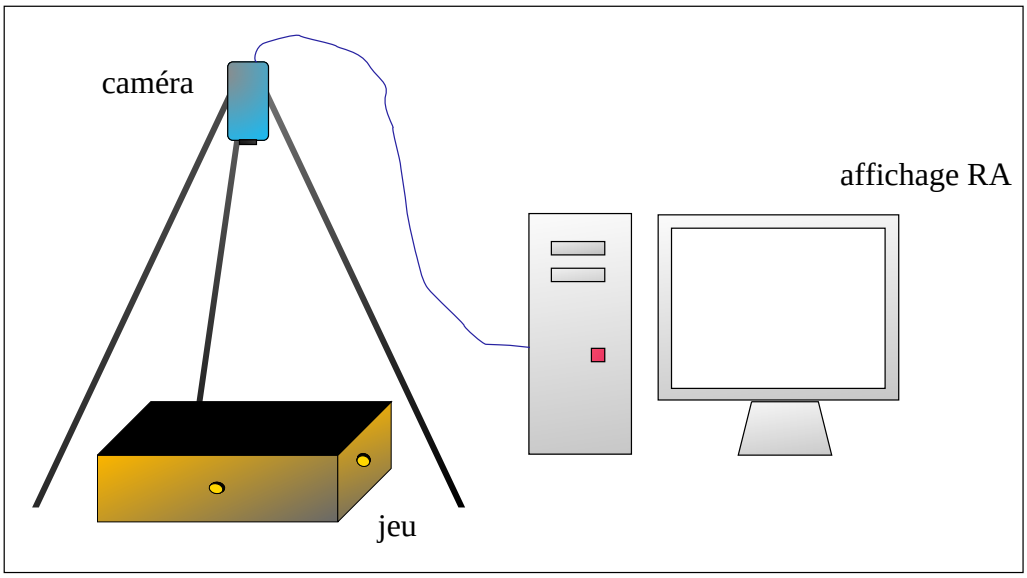


FIG. 1.15 – Dispositif de réalité augmentée : jeu du labyrinthe

le jeu du labyrinthe, ou le joueur, à l'aide de deux mollettes, fait effectuer des rotations en X et en Y à un plateau contenant un labyrinthe. Le but est de faire bouger le plateau de façon à faire parcourir le labyrinthe à une bille, en lui faisant éviter les obstacles, des trous.

En plaçant une caméra au dessus du jeu, et en faisant apprendre à l'algorithme le motif du labyrinthe, il est possible de connaître les déplacements du plateau. En appliquant un modèle physique simple pour gérer la vitesse/accélération, on peut alors faire se déplacer à l'écran une bille virtuelle dont les mouvements coïncident avec le mouvement du plateau.

La figure 1.15 montre le dispositif. Les utilisateurs de cette démo, après le léger temps d'adaptation nécessaire pour adapter ses mouvements à ce qui se passe à l'écran, ont pu jouer comme avec une vrai bille 1.16.

Pour rendre le jeu plus intéressant, un fichier texte contenant la position des murs et des trous a été créé en prenant des mesures sur le plateau réel.

FIG. 1.16 – Images de labyrinthe - haut : bille virtuelle dans le labyrinthe - bas : dispositif physique

Chapitre 2

Suivi planaire contour/texture

2.1 Introduction

La méthode de suivi proposée par Jurie et Dhome[36] et présentée au chapitre précédent est une méthode originale pour le suivi d'objets texturés. Elle permet de suivre le déplacement de motifs texturés avec une grande rapidité et une grande précision. Malheureusement, elle n'est pas applicable dans le cas de motifs peu texturés ou de couleur uniforme.

Or dans de nombreuses applications industrielles ce type de suivi peut avoir un intérêt, pour le contrôle de qualité sur des pièces manufacturées, par exemple. Certains motifs ne sont pas suffisamment texturés pour pouvoir être suivis par des algorithmes basés sur la luminance des motifs ; pour ces objets, il est plus indiqué de les suivre en utilisant leurs contours dans les images, alors que pour d'autres objets, c'est le contraire.

C'est pourquoi nous proposons ici une méthode hybride permettant de suivre des motifs visuels rigides en prenant en compte les caractéristiques les plus adaptées au motif à suivre.

2.1.1 Suivi de contour / suivi de texture

Comme nous l'avons vu au chapitre précédent, les algorithmes de suivi d'objets dans des images vidéo peuvent être vus comme des algorithmes d'optimisation. Il s'agit de déterminer des paramètres d'un vecteur d'état manière à optimiser le *recalage* du modèle sur une image. Ces paramètres d'état peuvent comporter des informations sur la position, l'orientation, les vitesses de l'objet et peuvent éventuellement décrire des changements d'aspect ou de propriétés des objets.

Une façon un peu différente de celle vue au chapitre précédent de classer les

algorithmes de suivi existants dans l'état de l'art est de les diviser en deux catégories, en fonction de la nature de la fonction à optimiser. Pour le premier type, il s'agit de minimiser des distances entre des primitives de l'image (points, lignes, b-splines, *etc.*) et des primitives d'un modèle de l'objet. Dans le second cas il s'agit de mesurer la ressemblance entre la fonction de luminance globale du motif et celle dans l'image étudiée.

Dans la première catégorie on trouve les travaux de Lowe [47], de Kollnig et Nagel [40] ou de Drummond et Cippola [23], qui déterminent la pose d'objets de modèles 3D connus.

Et dans la seconde on retrouve les travaux de Hager *et al.* [27] ou de Comaniciu *et al.* [13].

Notre approche, dans cette classification, se place à la frontière entre les deux catégories, le suivi d'éléments de contour appartenant plutôt à la première catégorie, tandis que l'utilisation d'informations de luminance pour le suivi de texture appartient plutôt à la seconde.

2.1.2 Contribution et plan du chapitre

A notre connaissance, peu de méthodes de suivi d'objet combinent différents types d'informations, tels que les contours et la texture. La contribution principale de ce chapitre réside justement dans la proposition d'une méthode hybride, capable de mélanger des informations de distances aux contours et des informations de texture, en s'adaptant ainsi au mieux aux différentes caractéristiques visuelles d'un objet.

Ce chapitre comporte trois parties. La section 2.2 présente les bases de la méthode proposée. Cette méthode repose sur l'utilisation de deux phases ; une phase dite d'apprentissage et une phase de suivi ; elles seront présentées en détail dans la section suivante. Enfin nous présenterons des résultats expérimentaux montrant la validité de l'approche.

2.2 Une méthode hybride pour le suivi de motif

Comme nous l'avons remarqué précédemment, pour suivre un motif visuel particulier il peut être plus intéressant d'utiliser des distances aux contours, alors que pour d'autres la prise en compte de la texture est préférable.

Nous proposons ici d'utiliser un critère mixte contour/texture. La méthode que nous proposons dans ce chapitre constitue une extension des travaux de Jurie et Dhome [36] présentée au chapitre précédent.

2.2.1 Représentation du motif

Comme expliqué précédemment, nous supposons que le motif planaire à suivre est composé des N points $(\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N)$, où $\mathbf{p}_i = (x,y)$, réunis dans le vecteur $\mathbf{P}$. Ces coordonnées, exprimées dans un repère lié au motif (différent du repère de l'image), sont supposées constantes.

Ces points sont choisis de manière à satisfaire deux contraintes :

– être bien répartis sur le motif. La répartition doit être homogène, les points ne doivent pas être tous sur une même zone tandis que le reste du motif reste vierge de points ;
– être placés dans des zones de l'image où le gradient spatial est fort, c'est à dire là où il y a de l'information.

En pratique, le motif est découpé en baquets, et un nombre minimal de points est choisi dans chaque baquet. Le choix se fait en tirant des points au hasard dans chaque baquet et en choisissant ceux ayant le gradient le plus fort.

Chacun des points ainsi sélectionnés est ensuite classé comme point de contour ou point de texture, à partir du critère suivant : soient x et y les coordonnées d'un point $\mathbf{p}$ de l'image, et $I(x,y) = I(\mathbf{p})$ l'intensité lumineuse en ce point. Nous définissons M comme :

$$M = \begin{pmatrix} \sum_w \frac{\partial I(\mathbf{p})}{\partial x}^2 & \sum_w \frac{\partial I(\mathbf{p})}{\partial x}\frac{\partial I(p)}{\partial y} \\ \sum_w \frac{\partial I(\mathbf{p})}{\partial x}\frac{\partial I(\mathbf{p})}{\partial y} & \sum_w \frac{\partial I(\mathbf{p})}{\partial y}^2 \end{pmatrix} \tag{2.1}$$

où w est une fenêtre carrée centrée sur le point. Soient α et β les deux valeurs propres de M. Alors, le point est considéré comme point de contour si $\alpha + \beta$ est grand et si, simultanément, $\alpha * \beta$ est faible. Sinon, il est considéré comme point de texture. Cela revient à dire que la fonction d'auto-corrélation de la surface n'est courbée que dans une seule direction. Ce critère est très proche du critère proposé par Harris et Stephens [28] pour la détection de coins.

A l'issue de cette étape, chaque point $\mathbf{p}_i$ du motif est classé comme point de contour ou point de texture, ce que nous notons $\mathbf{C}(\mathbf{p}_i) = 1$ et $\mathbf{C}(\mathbf{p}_i) = 0$, respectivement.

Le motif est caractérisé par un vecteur de forme $\mathbf{V} = (v_1, \ldots, v_N)$ où les v_i sont définis par :

$$v_i = \mathbf{V}(\mathbf{p}_i) = \begin{cases} I(\mathbf{p}_i, t) & \text{si } \mathbf{C}(v_i) = 0 \\ DI(\mathbf{p}_i, t, nx, ny) & \text{si } \mathbf{C}(v_i) = 1 \end{cases} \tag{2.2}$$

où $I(\mathbf{p}, t)$ représente le niveau de gris de l'image I au temps t. $DI(\mathbf{p}_i, t, nx, ny)$ représente une distance du point à un contour de l'image, mesurée dans l'image I

au temps t, dans une direction (nx, ny) perpendiculaire au contour lié au modèle. Cette distance est représentée figure 2.1.

Nous notons par la suite **V** sous la forme $V(\mathbf{P}, t)$ pour montrer le lien existant entre le vecteur de forme et, d'une part, les coordonnées des points du motif, et, d'autre part, le temps.

Le vecteur de forme est un ensemble de mesures décrivant le motif ; il contient d'une part l'intensité lumineuse de certains points de l'image et d'autre part des distances de points à des contours.

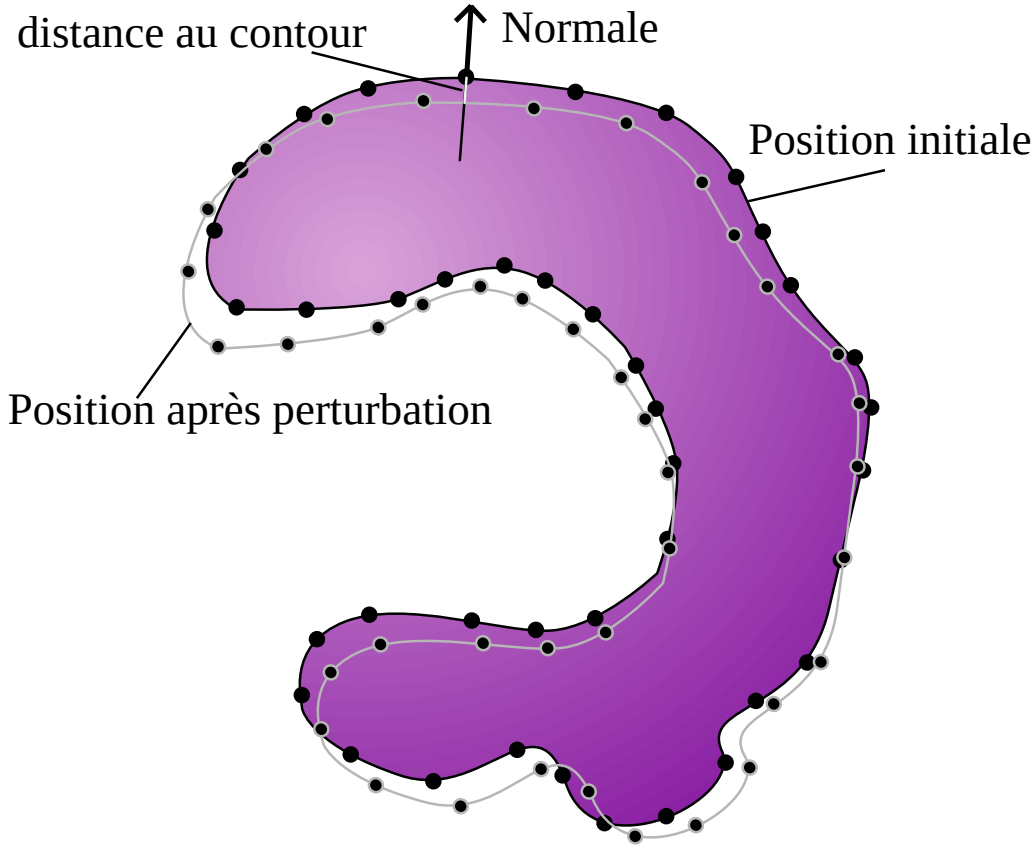


FIG. 2.1 – Distance du point au contour le plus proche

2.2.2 Représentation du mouvement

Il nous faut maintenant modéliser les déplacements du motif dans l'image.

Soit $\mu(t) = (\mu_1(t), \mu_2(t), \ldots, \mu_n(t))$ le vecteur des paramètres représentant la position du motif à l'instant t. Ces paramètres sont les paramètres d'une fonction de changement de repère, amenant un point $\mathbf{p}$ du repère *motif* au point $\mathbf{p}'$ dans l'*image* (figure 2.2). On considère qu'il existe une fonction f décrivant la position du motif, fonction telle que

$$\mathbf{p}' = f(\mathbf{p}, \mu(t)) \tag{2.3}$$

Nous n'introduisons pas de restriction sur cette fonction de mouvement. Elle peut représenter des transformations simples tels que des translations planaires, mais aussi des transformations plus complexes comme des homographies, ou des mouvements impliquant des déformations, même si cette dernière possibilité n'a pas été étudiée.

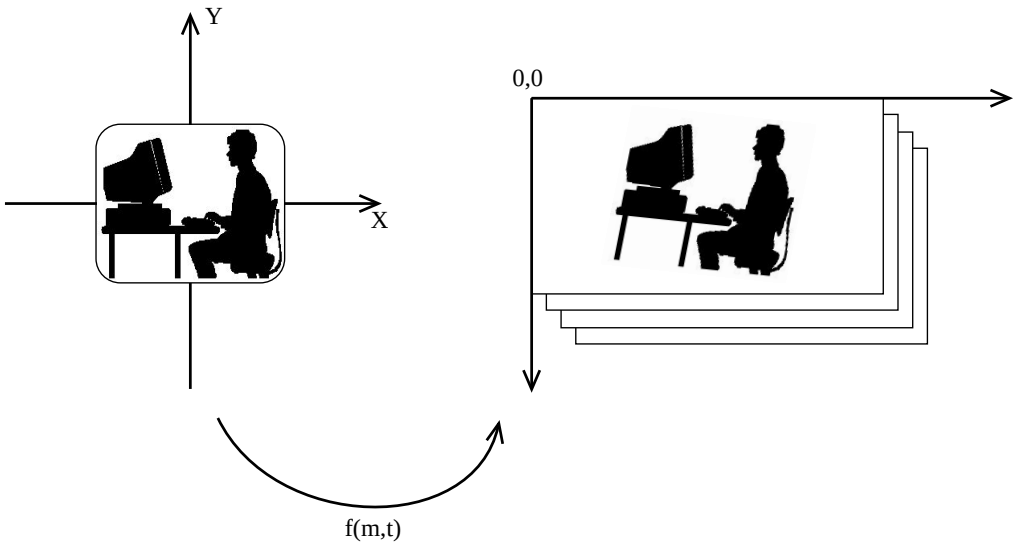


FIG. 2.2 – Repère image / repère motif

L'expression 2.3 signifie que le point $\mathbf{p}$ vient en $\mathbf{p}'$ si on lui applique le mouvement de paramètres μ. Nous notons également :

$$\mathbf{V}(\mu(t), t) = \mathbf{V}(f(\mathbf{P}, \mu(t)))$$

où $f(\mathbf{P}, \mu(t)) = (f(p_1, \mu(t)), \ldots, f(p_N, \mu(t)))$. Nous supposons que $\mathbf{V}$ est dérivable par rapport au temps.

Nous appelons par la suite *mouvement* de l'objet sa différence de position entre deux images, et nous notons $\delta\mu(t)$ ce mouvement. Ainsi, nous avons par définition :

$$\delta\mu(t) = \mu(t) - \mu(t-1)$$

Par la suite, la transformation réalisée est supposée est linéaire dans l'espace du plan projectif. Cela nous permet de représenter des déplacements 2D tels que translations, rotations planaires, affinités, mais aussi des mouvements 3D du plan au moyen d'homographies.

La transformation peut donc s'exprimer par un produit de matrices, après avoir écrit les coordonnées des points avec des coordonnées homogènes. Cette matrice est notée $\mathbf{F}(\mu(t))$. La relation (2.3) s'écrit donc

$$\mathbf{p}' = \mathbf{F}(\mu(t)) \times \mathbf{p} \tag{2.4}$$

où $\mathbf{p}$ et $\mathbf{p}'$ sont écrits en coordonnées homogènes.

2.2.3 Suivi du motif

En utilisant les définitions précédentes, nous proposons un suivi basé sur la minimisation d'un critère de type SSD[1]. Suivre l'objet revient à donner à chaque instant t sa position $\mu(t)$ telle que $\mu(t)$ minimise le critère :

$$\varepsilon(t) = \| \mathbf{V}(\mu_0, t_0) - \mathbf{V}(\mu(t), t) \| \tag{2.5}$$

où $\mu_0 = \mu(t_0)$ est la position du motif à la première image de la séquence. Nous supposons que μ_0 est connu à la première image de la séquence.

Nous recherchons une méthode itérative donnant la position $\mu(t)$ en fonction de la position $\mu(t-1)$ et de l'image à l'instant t. Le principe de base consiste à considérer qu'au voisinage d'une certaine position du contour suivi, la fonction permettant de passer des paramètres du mouvement effectué par le contour à la modification de l'image est linéaire.

En supposant que la position à l'instant $t-1$ soit parfaitement connue, c'est à dire que $\varepsilon_{t-1} = 0$,

$$\mathbf{V}(\mu_0, t_0) = \mathbf{V}(\mu(t-1), t-1) \tag{2.6}$$

$\mathbf{V}$ est une fonction dépendant de t et de μ, nous pouvons écrire son développement limité au premier ordre comme :

$$\begin{aligned} & \mathbf{V}(\mu(t-1) + \delta\mu, t-1+\delta t) = \mathbf{V}(\mu(t), t) \\ = \quad & \mathbf{V}(\mu(t-1), t-1) + (\mu(t) - \mu(t-1))\mathbf{V}_\mu(t) + \delta t \mathbf{V}_t(t), \end{aligned}$$

où $\mathbf{V}_\mu(t)$ représente la dérivée de $\mathbf{V}$ par rapport à μ et $\mathbf{V}_t(t)$ sa dérivée par rapport au temps, ces deux dérivées étant calculées au temps t. Nous supposons que le temps séparant deux images est par définition de 1.

[1] « Sum of Square Differences », *ie.* somme des différences au carré en français

En supposant que

$$\mathbf{V}_t(t) = -\mathbf{V}(\mu(t-1),t-1) + \mathbf{V}(\mu(t-1),t),$$

noté par la suite $\delta\mathbf{V}(t)$, en prenant en compte le fait que le minimum est obtenu pour $\bigtriangledown\varepsilon(t) = 0$ dans (2.5), et utilisant (2.6), nous obtenons :

$$\begin{aligned} 0 &= \mathbf{V}(\mu(t-1),t-1) - \mathbf{V}(\mu(t),t) \\ &= (\mu(t) - \mu(t-1))\mathbf{V}_\mu(t) - \delta\mathbf{V}(t) \end{aligned}$$

ce que nous pouvons également mettre sous la forme :

$$\mu(t) = \mu(t-1) + \mathbf{V}_\mu^{-1}(t)\delta\mathbf{V}(t)$$

ou encore :

$$\delta\mu(t) = \mathbf{V}_\mu^{-1}(t)\delta\mathbf{V}(t)$$

où $\delta\mu(t)$ représente le mouvement de l'objet à l'instant t, et $\mathbf{V}_\mu^{-1}(t)$ la matrice Jacobienne inverse. Cette relation nous permet donc de déduire le mouvement de l'objet à partir d'une variation du vecteur de forme. La matrice Jacobienne $\mathbf{V}_\mu(t)$ donne les variations des paramètres du vecteur de forme en fonction d'un mouvement $\delta\mu$ de l'objet. Il s'agit d'une matrice de taille $N \times n$ où n est le nombre de paramètres décrivant le mouvement, et N la dimension du vecteur de forme. La matrice Jacobienne n'est donc pas inversible. Nous avons montré, et c'est une contribution importante de notre approche, qu'une utilisation de la pseudo-inverse $\mathbf{V}_\mu^\dagger(t)$, au moyen de l'équation :

$$\delta\mu(t) = \mathbf{V}_\mu^\dagger(t)\delta\mathbf{V}(t), \tag{2.7}$$

donne des résultats bien moins intéressants que de calculer $\mathbf{H}(t)$ telle que :

$$\delta\mu(t) = \mathbf{H}(t)\delta\mathbf{V}(t), \tag{2.8}$$

par apprentissage direct.

La méthode de suivi peut être résumée de la façon suivante : nous commençons par une phase d'*apprentissage* durant laquelle nous déterminons une matrice H pour de petits déplacements du contour autour de la position initiale. Si le contour a déjà été appris, nous pouvons passer directement à la deuxième phase, le *suivi* en lui-même, où nous calculons la différence entre deux images de la séquence pour obtenir $\delta\mathbf{V}(t)$ puis nous en déduisons $\delta\mu(t)$ par l'équation (2.8).

On peut considérer que cette équation correspond à la modélisation de points dans un espace de dimension N par n hyperplans dont les coefficients $h_{i1}\dots h_{iN}$ peuvent s'écrire

$$\left\{\begin{array}{rcl} 0 & = & (h_{11},\dots,h_{1N},-1)(\delta i_1,\dots,\delta i_N,\delta\mu_1)^T \\ 0 & = & (h_{i1},\dots,h_{iN},-1)(\delta i_1,\dots,\delta i_N,\delta\mu_i)^T \\ 0 & = & (h_{1n},\dots,h_{nN},-1)(\delta i_n,\dots,\delta i_N,\delta\mu_n)^T \end{array}\right.$$

Bien évidemment il serait totalement inefficace de recalculer à chaque image la matrice **H**(t). En effet, le calcul d'optimisation impliqué dans cette phase est coûteux en temps de calcul.

La matrice **H** est calculée une seule fois, dans le repère du motif, et nous effectuons des changements de repère de façon à toujours nous trouver au voisinage de la position initiale. Nous développons ce point par la suite.

2.3 Les phases d'apprentissage et de suivi

Nous venons de voir que l'algorithme proposé repose sur deux phases : une phase d'apprentissage et une phase de suivi. Nous allons détailler chacune de ces deux phases et les modifications apportées par la prise en compte des contours.

Mais auparavant, nous allons expliquer comment la mesure du vecteur de forme est réalisée.

2.3.1 Une mesure relative du vecteur de forme

Nous avons en effet signalé que, pour éviter de calculer la matrice **H** pour chaque nouvelle position du motif dans l'image, nous proposons de la calculer dans le repère initial du motif.

Ainsi, pour calculer le vecteur de forme, il faudrait appliquer $f^{-1}(\mu)$ à l'image à traiter, calculer le vecteur de forme sur cette image, en déduire le mouvement $\delta\mu$ correspondant. Appliquer la transformation $f^{-1}(\mu)$ à l'image entière est un travail coûteux et inutile. Coûteux car il est nécessaire de traiter l'ensemble des points de l'image, et inutile car seul un nombre très limité d'informations seront utilisées : la mesure de la luminance en certains points, et la distance au contour le plus proche dans les autres cas. C'est pourquoi nous proposons de procéder différemment.

A l'issue de l'étape de modélisation du motif, chaque point $\mathbf{p}_i$ est classé comme point de contour ou point de texture ($\mathbf{C}(\mathbf{p}_i) = 1$ ou $\mathbf{C}(\mathbf{p}_i) = 0$, respectivement).

Le motif est caractérisé par un vecteur de forme $\mathbf{V} = (v_1,\dots,v_N)$ où les v_i sont définis par l'équation (2.2).

Ainsi, pour faire la mesure $\mathbf{V}(\mathbf{p}_i)$ dans l'image, le point se trouvant à la position $f(\mathbf{p}_i)$, si $\mathbf{C}(v_i) = 0$ on prend $v_i = I(f(\mathbf{p}_i, \mu), t)$ et si $\mathbf{C}(v_i) = 1$ on prend la distance depuis le point $f(\mathbf{p}_i)$ jusqu'au contour image le plus proche, dans la direction de la normale au contour liée au modèle.

La direction de la normale au contour est mémorisée à l'aide d'un point $\mathbf{n} = (n_x, n_y)^t$: la normale en $\mathbf{p}$ est définie comme la droite passant par $\mathbf{p}$ et $\mathbf{n}$. La transformation f étant représentée par la matrice $\mathbf{F}$ (comme expliqué section 2.2.2), la direction de la normale est donnée par les points $\mathbf{p}' = \mathbf{F} \times \mathbf{p}$ et $\mathbf{n}' = \mathbf{F} \times \mathbf{n}$.

L'étendue de cette zone d'exploration est déterminée par l'étendue de la zone d'apprentissage. Par exemple, si l'on choisit de perturber le motif jusqu'à 20% de sa taille, la zone d'exploration est choisie de telle façon qu'un déplacement de 20% puisse être retrouvé. Dans les expériences exposées dans ce chapitre, la zone d'exploration est indexée sur la longueur de la "diagonale" du quadrilatère suivi. Pour éviter des problèmes d'échelle, on ajoute un terme qui prend en compte le déplacement du motif. Ainsi, si l'on appelle l_{zone} la longueur de cette zone et $d(\mathbf{p}', \mathbf{n}')$ la longueur de la normale au moment considéré, on a

$$l_{zone} = d(\mathbf{p}', \mathbf{n}') \times l_{apprentissage} \times l_{diagonale}$$

Nous mesurons alors la distance d au contour le plus proche, dans l'image, et obtenons directement la distance dans le repère du modèle, en remarquant que la distance D dans le repère modèle est $\frac{d}{\|\mathbf{n}'\|}$.

Ainsi, avec une quantité de calcul très minime, nous obtenons le vecteur de forme que nous obtiendrions si nous appliquions la transformation f^{-1} à l'image.

2.3.2 Apprentissage

Comme nous l'avons dit ci-dessus, il faut éviter à tout prix de devoir recalculer $H(t)$ à chaque nouvelle image. Il est possible de supposer que $\mathbf{H}$ est constante en établissant la relation (2.8) dans le repère lié au motif et non pas dans le repère lié à l'image. Ainsi, dans la phase de suivi, il suffira de ramener l'image dans le repère du motif pour pouvoir utiliser la relation (2.8). C'est pour cette raison que nous avons dû supposer que f était inversible.

L'estimation de la matrice $\mathbf{H}$ se fera donc dans un voisinage de la fonction *identité*. Pour ceci, effectuons N_{Pert} variations de position aléatoires $\delta\mu^i$ et obtenons les N_{Pert} valeurs de $\delta\mathbf{V}^i$ correspondantes. Les mouvements $\delta\mu^i$ sont stockés dans une matrice $\mathbf{DMU}_{N_{Pert} \times n}$ et les vecteurs de forme dans une matrice $\mathbf{DI}_{N_{Pert} \times N_{Pt}}$.

Il suffit ensuite de calculer la "meilleure" matrice **H**, c'est-à-dire la matrice pour laquelle la somme

$$\sum_{i=1}^{i \leq N_{Pert}} (\delta\mu^i - \mathbf{H} \times \delta V^i)^2$$

est la plus faible possible. Cela revient à minimiser un système au sens des moindres carré, ce qui peut être fait au moyen de la relation suivante :

$$\mathbf{H} = \mathbf{DMU}.\mathbf{DI}^{\dagger}$$

où $\mathbf{DI}^{\dagger}$ représente la pseudo-inverse de **DI**.

H, résultat de notre apprentissage, est une matrice qui va nous permettre de retrouver le mouvement correspondant à une différence de vecteur de forme (différence de texture et distance au contour) trouvée.

Les données sont obtenues lors de l'apprentissage de la même façon qu'elles le seront lors du suivi. On suppose ainsi que si l'algorithme effectue une erreur en localisant un point pendant l'une des phases, il reproduira cette erreur lors de la seconde. Cela permet de minorer les problèmes dus à de mauvaises mises en correspondances des points de contour.

2.3.3 Suivi

Une fois la matrice **H** connue, nous pouvons passer à la phase de suivi proprement dite.

Lors du suivi, nous n'utilisons pas d'étape de prédiction de mouvement. Il est supposé que dans l'image suivante la position du motif sera la même qu'à l'image actuelle. Pour que le programme de suivi fonctionne correctement, il faut donc que le déplacement du contour entre deux images soit assez faible pour que la "prédiction" soit correcte. En pratique, le domaine de convergence est suffisamment large pour que le motif soit suivi malgré des amplitudes de déplacement importantes.

Nous construisons le vecteur de forme en mesurant dans l'image, pour chaque point du motif, le niveau de gris du point ou la distance au contour (en fonction du type de point). Nous stockons la différence entre le vecteur ainsi formé et le vecteur de forme obtenu pour un mouvement nul dans un vecteur δV. La construction de ce vecteur de forme se fait dans le repère lié au motif, comme expliqué section 2.3.1.

Après une multiplication de ce vecteur avec la matrice **H**, nous obtenons un vecteur $\delta\mu$ correspondant à l'estimation des paramètres de mouvement, dans le repère du motif. Toutefois cette matrice ne permet de connaître le déplacement que pour un objet se situant au voisinage de sa position initiale. Pour pouvoir continuer

d'appliquer l'algorithme même après un changement de position du motif, Nous devons ramener le mouvement dans le repère de l'image. Nous obtenons ainsi le mouvement

$$f(\mu(t)) = f(\mu(t-1)) \circ f(\delta\mu(t))$$

Dans le cas où le déplacement f est modélisé par une transformation linéaire dans le plan projectif, $f(\mu(t))$ est représenté par une matrice $\mathbf{F}(\mu(t))$, dont l'estimation est :

$$\mathbf{F}(\mu(t)) = \mathbf{F}(\mu(t-1)) \circ \mathbf{F}(\delta\mu(t))$$

Notre algorithme de suivi peut s'écrire en pseudo-langage de la façon suivante, en appelant *mu* le déplacement global jusqu'à l'image *n* et *pt* les points du contour sur l'image initiale.

On constate que la complexité algorithmique du calcul du vecteur de mesure est plus élevée pour les points de contour que pour les points de texture. Si le nombre de pas dans la zone d'exploration est élevé, le temps de traitement pour chaque image est d'autant plus important.

Il ne faut pas oublier de prendre en compte l'affichage et les temps d'accès à la caméra et à la mémoire, qui ont une grande importance dans les temps d'exécution. Toutefois cela reste relativement raisonnable, et l'exécution de l'algorithme s'effectue bien en temps réel vidéo (25 images/s).

Il est facile de constater que la complexité algorithmique est faible. Pour chaque image, comptons le nombre de multiplications, qui sont les opérations les plus coûteuses en temps processeur. Il y a N_{Pt} points étudiés, et sur la normale nous parcourons N_{norm} valeurs. A l'intérieur de la boucle la plus interne, nous trouvons 4 multiplications, soit au total $4 * N_{Pt} * N_{norm}$ multiplications. A cela on ajoute la complexité du calcul matriciel : $n * N_{Pt}$ (avec n le nombre de paramètres de μ). Soit un total de

$$C = N_{Pt}.(4 * N_{norm} + n) \tag{2.9}$$

$$C = O(N_{Pt} * N_{norm}) \tag{2.10}$$

Pour chiffrer un peu ces résultats, disons qu'il faut approximativement plusieurs dizaines de milliers de cycles pour traiter une image. Les processeurs actuels effectuent un, voire deux milliards de cycles à la secondes. Cela signifie que théoriquement nous pourrions traiter plusieurs milliers d'images par seconde...

2.4 Validation expérimentale

Nous avons implanté l'algorithme proposé sur un ordinateur de type PC, équipé d'une caméra numérique SONY délivrant des images en temps réel vidéo, au

```
% -- changement de référentiel
% pour les points
PTI ← F(MU)* PT
% pour les normales
NORMI ← F(MU)*NORM

% -- recherche des nouvelles positions
pour I de 1 à N_PT faire
    si point de contour
       DNI ← NORMI - PTI
       GRADMAX ← 0
       pour K le long de la normale
          P1 ← PT(I) + (K-1) * DNI
          P2 ← PT(I) + (K+1) * DNI
          GRAD = Image(P2) - Image(P1)
          si GRAD > GRADMAX
             DI(I) ← K
             GRADMAX ← GRAD
          finsi
       finpour
    sinon si point texture
       DI(I) ← Image(PT(I))
    finsi
finpour

% -- calcul du déplacement
DMU ← H * DI

% -- calcul du déplacement global
F(MU) ← F(MU) * F(DMU)
```

TAB. 2.1 – Algorithme de suivi hybride

moyen d'une communication FireWire (IEEE 1394).

FIG. 2.3 – Exemples de suivi de motifs texturés (1)

Les motifs sont représentés à l'aide d'une centaine de points. La transformation utilisée est une homographie du plan projectif. Les figures 2.3 et 2.4 présentent des images issues du traitement de séquences en temps réel.

Pour démontrer l'apport de la composante de contour, nous avons utilisé l'algorithme de suivi hybride sur une séquence générée artificiellement. Cette séquence contient un motif non texturé se déplaçant rapidement. Une image extraite de cette séquence est visible à la figure 2.5.

Nous avons suivi le motif de cette séquence en n'utilisant que des points de

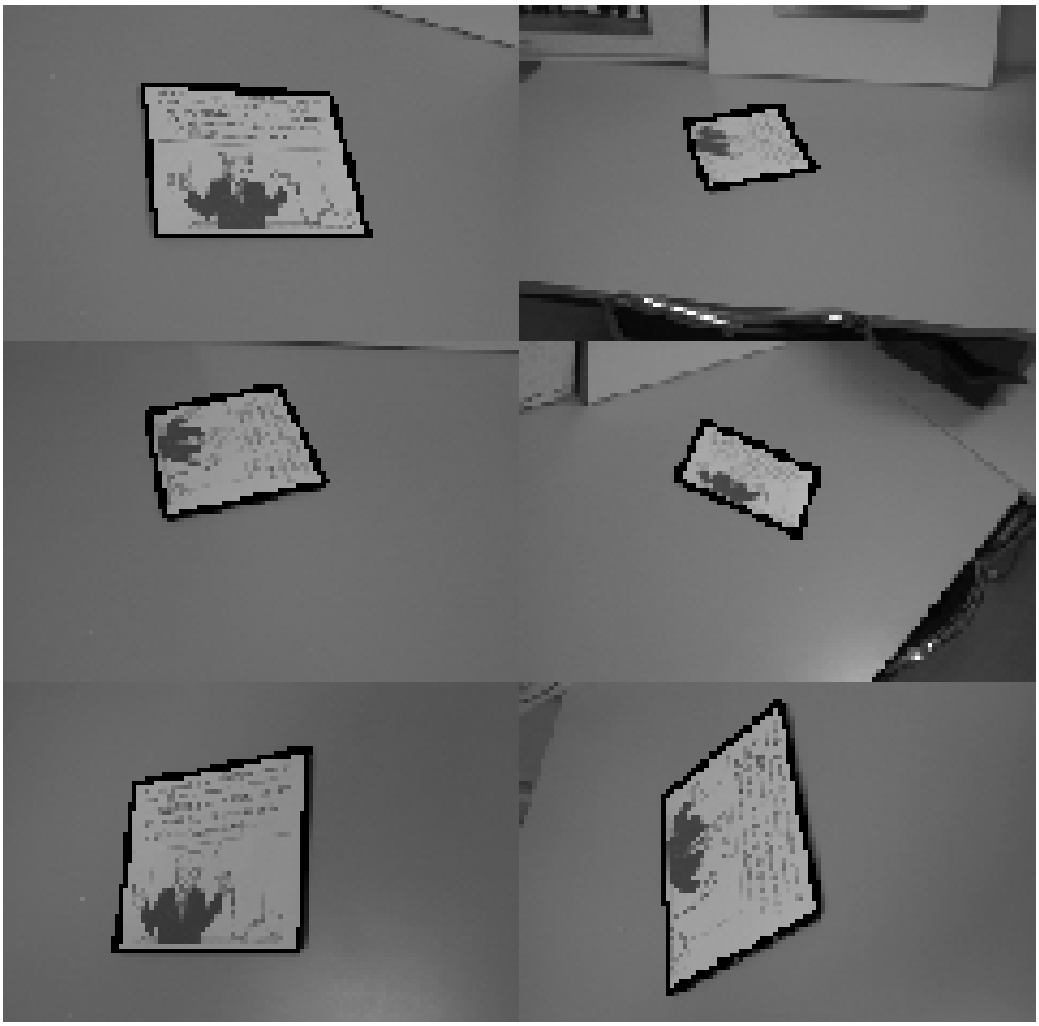


FIG. 2.4 – Exemples de suivi de motifs texturés (2)

texture, puis uniquement des points de contour, et enfin avec notre méthode hybride. La méthode hybride a sélectionné 80 points de contour et 20 points de texture. Cette prédominance des points de contour sur les points de texture est due au fait que le motif n'est pas "texturé".

La figure 2.6 montre l'estimation de la position de l'un des coins de la zone suivie pour chacun des algorithmes, ainsi que la position réelle.

Nous constatons que l'algorithme de suivi n'utilisant que des points de texture décroche au bout de 28 itérations, tandis que le suivi à base de points de contour suit le motif jusqu'à la fin de la séquence. Nous constatons aussi que la préci-

	contour	texture	hybride
temps d'exécution	16.18 ms	9.53 ms	14.14 ms

TAB. 2.2 – Temps d'exécution suivant le type de suivi utilisé

sion de l'algorithme hybride n'est pas subpixellique, contrairement à l'algorithme n'utilisant que la texture, du fait de la plus faible précision de la méthode de suivi de contour employée.

Un autre point important est le temps d'exécution. Étant donné que la complexité algorithmique du suivi avec des points de contour est plus élevée, le temps d'exécution est plus important si l'algorithme utilise de nombreux points de contour.

Dans le tableau suivant qui montre, dans des conditions similaires, le temps de traitement moyen d'une image suivant l'algorithme utilisé, nous constatons bien une baisse des performances avec l'utilisation de points de contour.

Nous avons aussi testé la taille du domaine de convergence du suivi pour un motif donné. Pour cela, nous avons fait apprendre le même motif à chacun des algorithmes et nous avons testé l'étendue des variations permises avant que l'algorithme ne perde le motif.

Les zones en noir de la figure 2.6 montre les translations pour lesquelles l'algorithme converge vers la bonne solution.

La figure 2.5 montre l'apport de la méthode par rapport au suivi classique. Le motif observé est une boîte en carton peu texturée mais présentant en certaines zones des contours forts. C'est un motif difficile à suivre du fait de la présence de zones similaires. La zone de convergence est observée pour des translations suivant x et y de plus ou moins 30%. Les graphiques représentent l'erreur obtenue entre le déplacement effectué et le déplacement effectivement trouvé.

Le suivi des points de texture est moyennement efficace dans cette configuration (graphique de gauche), tandis qu'avec les points de contour uniquement le résultat est plus mauvais (graphique du milieu). C'est avec la méthode hybride que la zone de convergence est la plus importante.

La zone de convergence obtenue avec des points de contour présente des discontinuités (les "blancs"), très visibles dans le cas des contours seuls. On peut voir, de plus, que les performances ne sont pas uniformes sur tout le domaine de convergence : la zone de convergence n'est pas symétrique en x et y. C'est dû au fait que le motif présente des problèmes de régularité : l'algorithme confond certaines dispositions du motif.

L'algorithme de suivi a choisi 45 points de texture et 55 points de contour pour ce motif. On peut constater sur ces graphiques l'apport de l'utilisation des deux

méthodes : le suivi de texture donne une zone de convergence restreinte mais sans discontinuités, tandis que le suivi de contour donne une zone de convergence plus étendue mais aussi plus irrégulière.

Nous pouvons donc dire que si nous avons réussi à étendre le domaine de convergence de notre algorithme à des motifs non texturés, ceci se fait au prix d'une augmentation du temps d'exécution et d'une légère baisse de la précision.

2.5 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle méthode de suivi de motifs hybride texture / contours.

L'idée consiste à représenter le motif au moyen d'un nombre limité de points. Les points sont choisis de manière à être répartis sur le motif et à se trouver dans des zones où la variation de la luminance est importante. Ces points sont ensuite classés comme point de contour ou point de texture. Un vecteur de forme contenant l'intensité du motif (pour les point de texture) et la distance au contour le plus proche (pour les points de contour) est ensuite calculé.

Lorsque le motif se déplace, une différence dans le vecteur de forme apparaît, et c'est cette différence qui est utilisée pour déduire le déplacement de l'objet. Cet algorithme permet de suivre un motif rigide en l'absence d'occultations.

Cette méthode, basée sur un algorithme connu pour son efficacité et sa robustesse, permet d'élargir le domaine d'utilisation de cet algorithme à des motifs pas ou peu texturés. Cette extension ne provoque qu'une faible augmentation du temps d'exécution.

FIG. 2.5 – Séquence de test générée artificiellement (images 5 et 15)

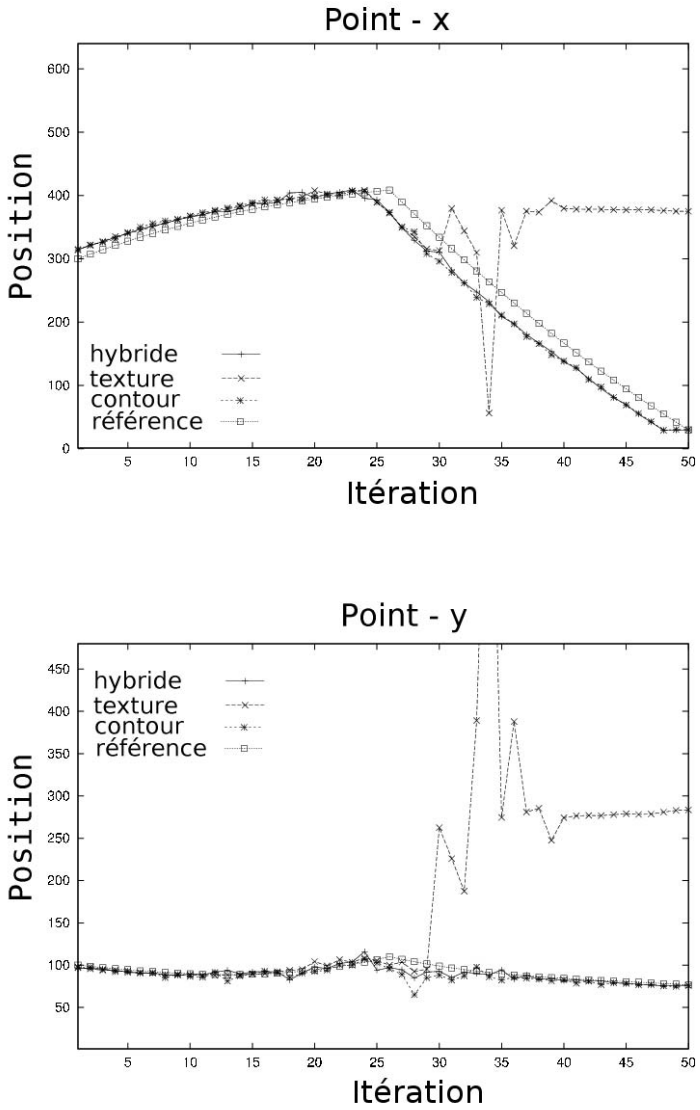


FIG. 2.6 – Estimation des coordonnées du coin supérieur droit du motif par les différents algorithmes

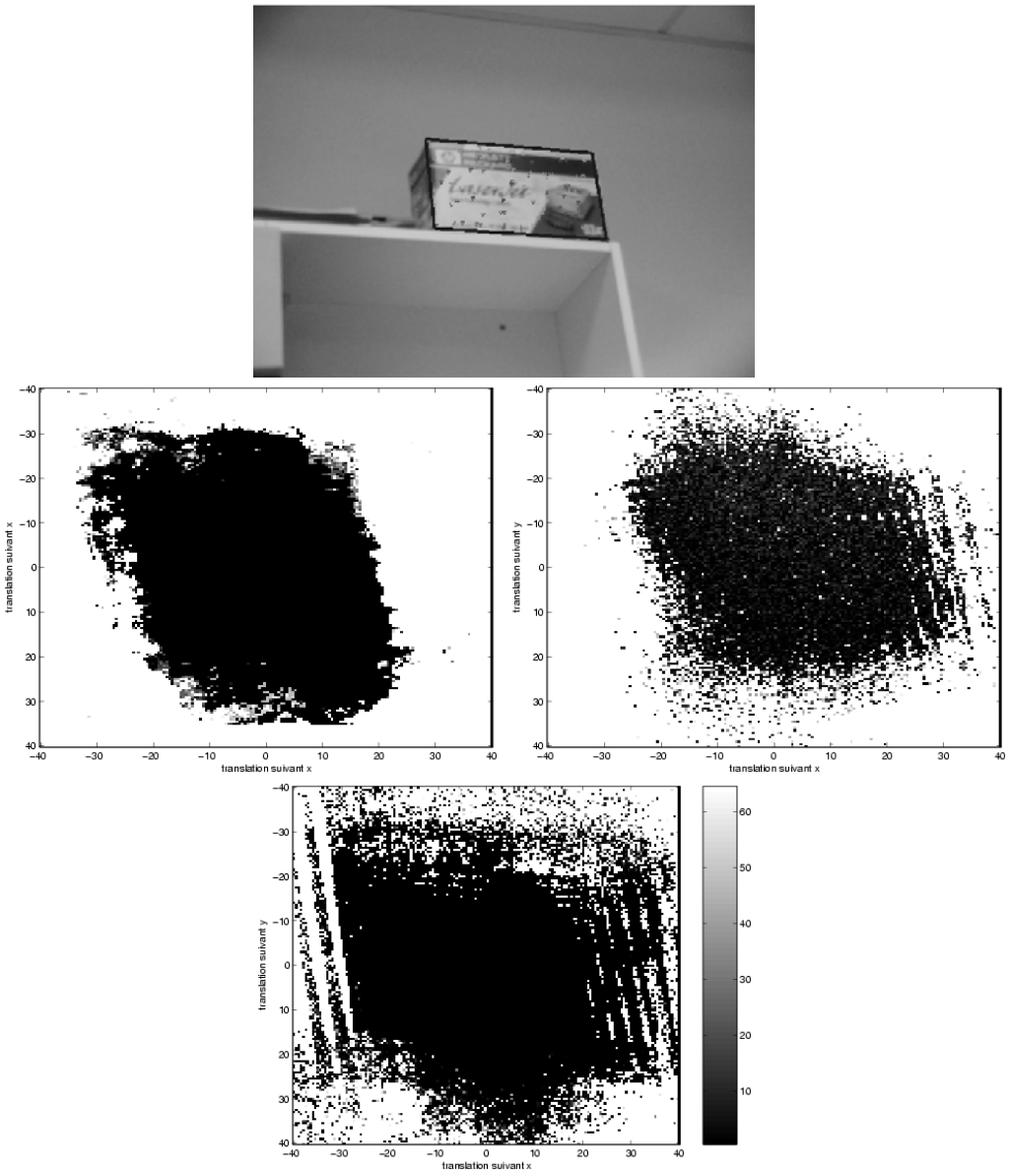


FIG. 2.7 – Zones de convergence pour des translations suivant x et y. haut : objet suivi, gauche : uniquement des textures, droite : uniquement des points de contour, bas : méthode hybride

Chapitre 3

Suivi rigide avec modèle 3D

3.1 Introduction

Les résultats du suivi de surfaces planes, tels que présentés en introduction de ce document et dans le chapitre précédent, montrent qu'il est relativement facile et peu coûteux de suivre des plans. Mais qu'en est-il des objets non plans ? Ces objets présentent des surfaces invisibles ou cachées, et s'auto-occultent, ce qui rend leur suivi plus difficile.

Le but des travaux décrits dans ce chapitre a été de prouver qu'il était possible d'utiliser les résultats du suivi de surfaces planes pour le suivi d'objets 3D. Cela est réalisé en utilisant de multiples occurrences de l'algorithme de suivi, réparties sur toute la surface de l'objet, et en utilisant un algorithme de calcul de pose robuste pour déterminer la pose de l'objet à partir des résultats de mesures fournies par les trackers[1] locaux.

3.2 Algorithme de suivi 3D développé

L'objectif de l'algorithme proposé est d'estimer la pose de l'objet-cible – notée **L**, et correspondant aux translations et rotations de l'objet – dans le système référentiel de la caméra à partir des images issues du flux vidéo. Une prédiction de **L** est supposée disponible pour chaque nouvelle image. Comme nous l'avons vu au chapitre 1, il existe différentes méthodes pour obtenir une telle prédiction. Il est possible d'utiliser un filtre de Kalman [39, 69], un filtre à particule [3], ou juste d'estimer que l'objet-cible se déplace suffisamment peu d'une image à l'autre

[1] on parle aussi d'occurrences de l'algorithme de suivi planaire.

pour que la position prédite à l'instant t soit la position de l'objet à l'instant $t-1$ (ce que nous faisons ici).

L'initialisation est faite par l'intermédiaire d'un algorithme de détection simple à base d'appariements de points d'intéret et de descripteurs SIFT [48]. Cela n'est utilisé que pour l'initialisation et dans le cas où l'objet-cible est « perdu » en cours de séquence, car les algorithmes de détection sont bien souvent trop lents pour pouvoir être utilisés en suivi temps-réel, même si c'est un peu moins vrai ces dernières années. Un article récent de Vincent Lepetit*et al.* [44] a d'ailleurs montré qu'il était possible de suivre un objet en utilisant des techniques de reconnaissance couplées à un algorithme de classification efficace.

La surface de l'objet-cible est décomposée en petites surfaces carrées de centre et de normale connus. Ces surfaces carrées, qui sont planes et tangentes à la surface de l'objet, sont appelées des *vignettes*[2]. Ces vignettes sont disposées sur le modèle 3D de l'objet. C'est leur nombre et leur répartition sur toute la surface de l'objet qui permet à l'algorithme d'être robuste aux occultations : un algorithme d'estimation de pose robuste utilise les résultats du suivi pour calculer la pose à partir des vignettes. Quand l'objet est partiellement occulté, seules les vignettes apportant une information pertinente sont utilisées pour calculer la pose.

Plusieurs vues-clefs sont nécessaires afin d'apprendre la texture de chacune des vignettes. Il faut en effet connaître la texture sur l'ensemble de la sphère de vues de l'objet pour pouvoir le suivre quelle que soit sa pose. La pose de l'objet dans ces vues est donnée manuellement en appariant certains points du modèle avec leur projection dans l'image-clef. Les vignettes peuvent être réparties de façon homogène sur toute la surface ou centrées sur des points d'intérêt suivant le critère de Harris et Stephens [28], les vignettes étant placées sur les points où ce critère est maximum. Un résultat typique de cette sélection de la position des patches peut être observé à la figure 3.1.

Pendant le suivi, la pose de l'objet est calculée en mettant en correspondance les centres des vignettes avec leur projection dans l'image. Ces appariements 3D-2D sont utilisés comme point de départ d'un algorithme d'estimation robuste de pose.

Il est possible de diviser la description de l'algorithme en deux parties : la mise en correspondance 3D-2D et l'estimation robuste de pose. Ces deux parties vont être décrites dans les deux sections qui suivent. Des résultats expérimentaux seront présentés dans la dernière section.

[2] ou « patchs » en anglais

FIG. 3.1 – Exemple de répartition des patches à la surface d'un cube, répartition en fonction des points d'intérêt.

3.3 Appariements 3D-2D

Ces appariements sont obtenus en utilisant de multiples occurrences de l'algorithme de suivi décrit dans le chapitre 1.

Plutôt que de déterminer les paramètres d'une homographie, soit huit paramètres, ces trackers locaux permettent de retrouver quatre paramètres (translations, facteur d'échelle et rotation). En effet, étant donné que seules les coordonnées du centre de la vignette sont recherchées, et comme l'algorithme de calcul de pose est robuste, la caractérisation d'une transformation homographique n'était pas nécessaire.

Les vignettes permettent donc d'estimer une transformation plane décrite par quatre paramètres, les translations en X et Y, la rotation suivant Z et le facteur d'échelle (figure 3.2).

Comme pour l'homographie, qui était décrite par les coordonnées des quatre coins d'un carré ayant subi la transformation, cette transformation est décrite par

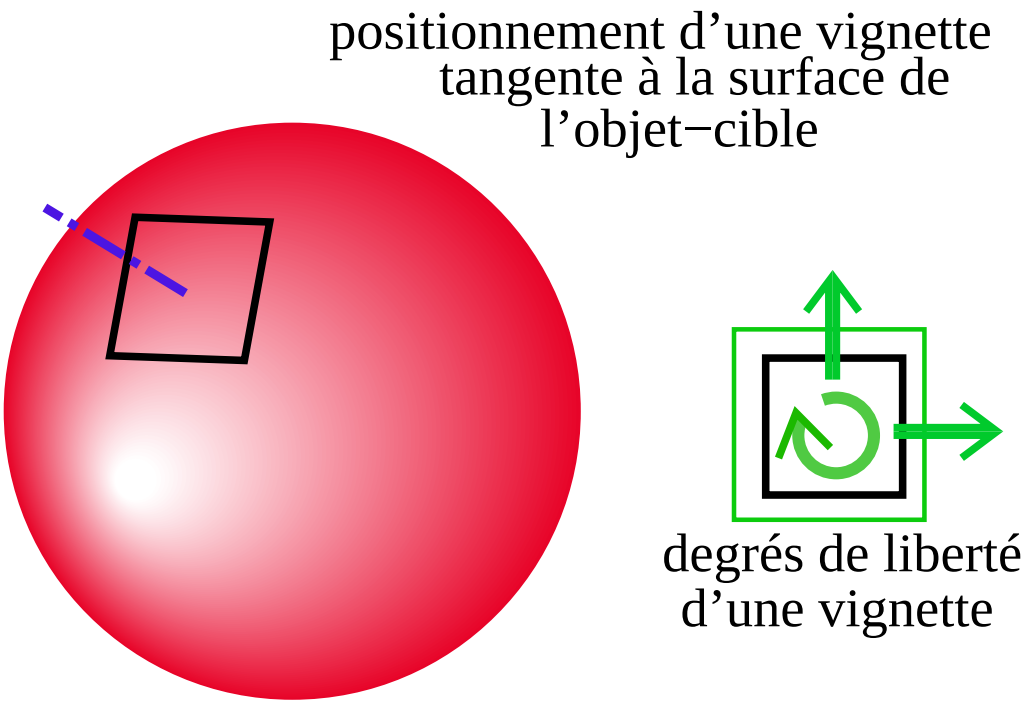


FIG. 3.2 – Les vignettes, des algorithmes de suivi plans à quatre degrés de liberté

les coordonnées de deux coins (soit quatre paramètres).

Le graphique de la figure 3.3 montre la précision obtenue par les vignettes lors du suivi, sur des petits déplacements (inférieurs à 10 pixels entre chaque image de la séquence). On peut constater que la distance entre la position du centre de la vignette attendu et la position trouvée ne dépasse pas 0.25 pixels.

La figure 3.4 montre plusieurs images d'une séquence de suivi réalisée avec deux vignettes indépendantes. Les vignettes continuent de suivre le motif malgrès des mouvement importants, la séquence ayant été prise « caméra à la main » et présentant par conséquent un certain nombre de tremblements et de mouvements brusques.

Un tel résultat est obtenu en réglant finement les paramètres d'apprentissage. Il y a deux paramètres principaux : le nombre d'échelles de niveau (qui correspond au nombre de matrice $\mathbf{H}$ apprises, voir éq. 1.1), et la valeur de ces niveaux. La figure 3.5 montre combien ces paramètres influent sur la taille du domaine de convergence. Ces graphes montrent les translations autorisées pour les vignettes en fonction des valeurs des échelles de niveau. Celles-ci se comprennent de la façon suivante : 10.0/5.0/1.0 signifie trois niveaux, 10.0, 5.0 et 1.0 donnent la

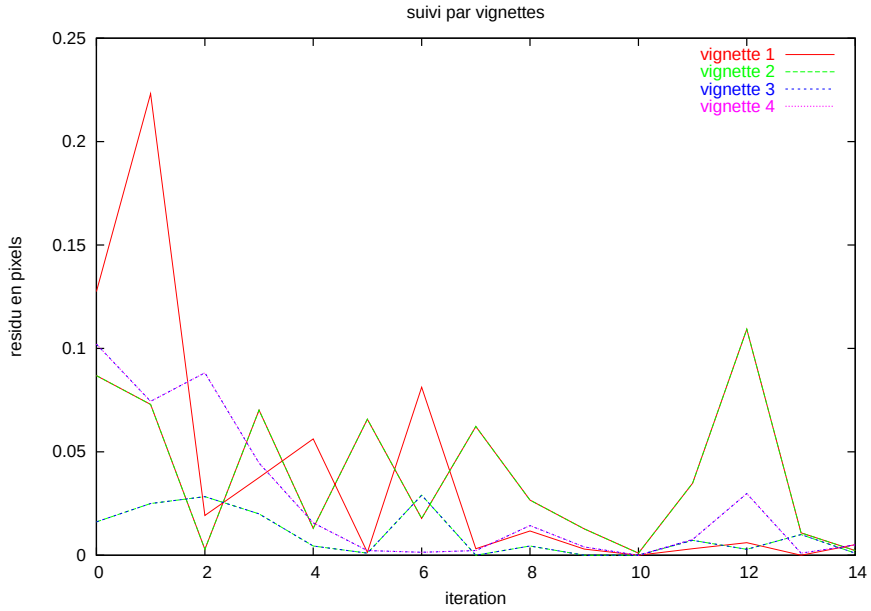


FIG. 3.3 – Précision du suivi

valeur de ces niveaux. 10.0 signifie que l'on apprend des déplacements sur une zone dont la taille est 10 fois celle de la vignette.

Ces graphes montrent que plus on augmente la taille de la zone d'apprentissage, plus les vignettes pourront tolérer de grosses translations, qui sont les transformations critiques. Mais passé un certain seuil, le domaine de convergence n'augmente plus, et au contraire un certain nombre de vignettes ont des domaines de convergence médiocres (graphe du bas).

Prises séparément, les vignettes ne sont pas robustes aux occultations, et ont une zone de convergence assez faible, de l'ordre d'une quinzaine de pixels. Chacune va fournir un appariement 2D/3D (2D : son centre ; 3D : sa position dans le référentiel de l'objet).

3.4 Estimation de la pose de l'objet

Une fois que les appariements 3D-2D ont été obtenus grâce au suivi planaire des vignettes, il est possible de calculer la pose de l'objet-cible. Cette section

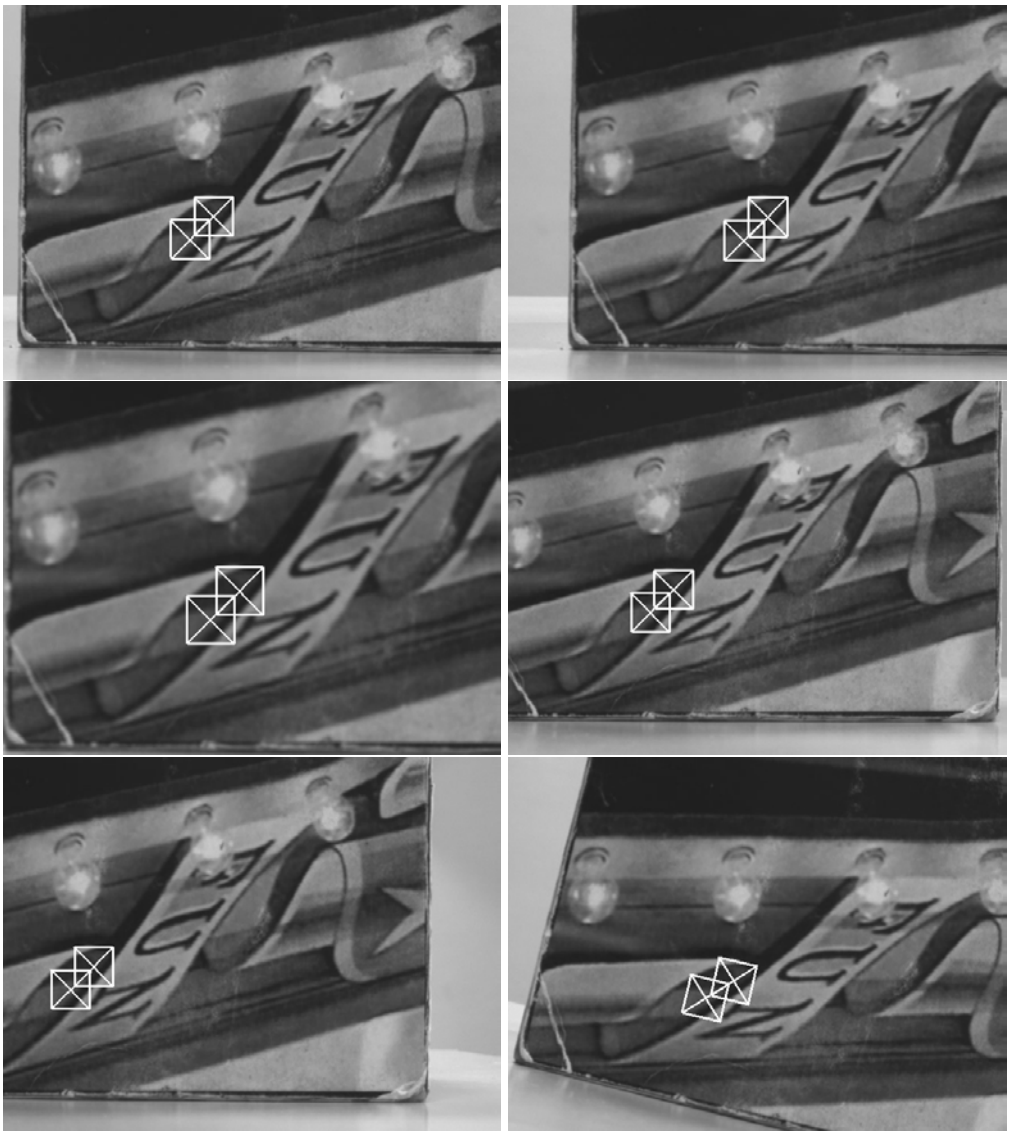


FIG. 3.4 – Séquence de test de suivi des vignettes

décrit comment la calculer à l'aide d'un estimateur robuste.

Pour chaque vignette, la phase d'appariement (section 2) nous donne les paramètres de déplacement de la vignette (rotation 2D, translations, facteur d'échelle) dans le référentiel de la vignette. Ces paramètres permettent de calculer la position du centre de chaque vignette dans l'image, ce qui nous donne les appariements 3D-2D. Ces appariements sont notés (x,y,X,Y,Z), avec (x,y) la position du centre de la vignette dans l'image et (X,Y,Z) les coordonnées 3D du centre de

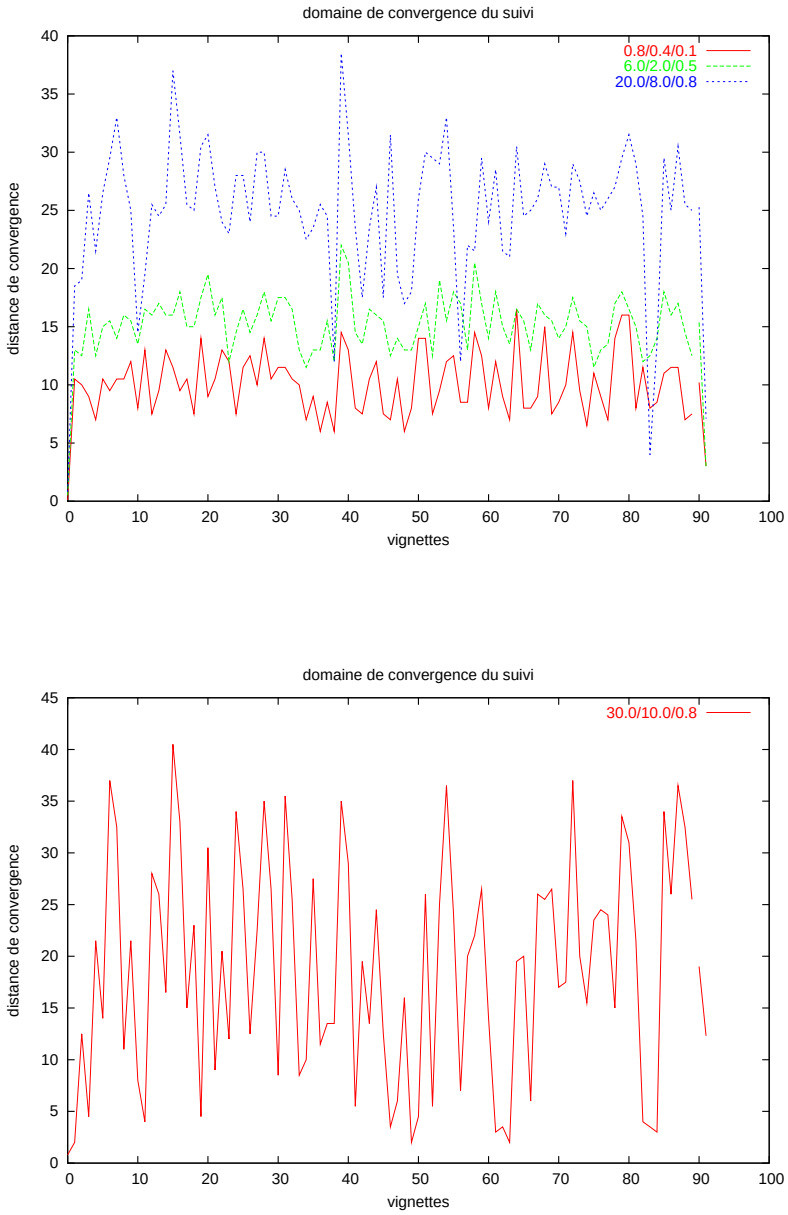


FIG. 3.5 – Domaine de convergence des vignettes

la vignette dans le référentiel de l'objet.

3.4.1 Algorithme théorique

La pose de l'objet est calculée est utilisant l'algorithme itératif développé par Lowe [45] et modifié par Dhome *et al.* [22]. Cet algorithme est rapide et précis ; moins de 10 itérations sont généralement nécessaires pour estimer la pose. C'est pourquoi cet algorithme est bien adapté à notre application. En pratique, quand une prédiction de la pose de l'objet est connue, la convergence est obtenue en moins de 3 ou 4 itérations.

Malheureusement, l'algorithme de Lowe n'est pas robuste aux faux appariements. Comme nous voulons que notre algorithme puisse supporter les occultations, il est nécessaire de supprimer les faux appariements – ou du moins de ne pas les prendre en compte – lors de la phase de calcul de pose. Cela peut être fait en ajoutant un M-estimateur de Tuckey [29] à l'algorithme de calcul de pose.

Soit $P_{i,t} = (X,Y,Z)$ les coordonnées 3D du point i à l'instant t, et $P_{i,t+1}$ ses nouvelles coordonnées à l'instant $t+1$. L'algorithme itératif de Lowe calcule les deux matrices $R_{\alpha,\beta,\gamma}$ et $T_{u,v,w}$ qui vérifient

$$P_{i,t+1} = R_{\alpha,\beta,\gamma}.P_{i,t} + T_{u,v,w}$$

Soit $p_{i,t+1} = (x,y)$ les coordonnées dans l'image du point $P_{i,t+1}$. Considérons $p_{i,t+1}$ comme l'intersection du plan de l'image avec deux plans d'« interprétation », quasi-horizontal et quasi-vertical (voir figure 3.6), définis par les vecteurs normaux $N^1_{i,t+1}$ et $N^2_{i,t+1}$ [3].

L'algorithme de Lowe minimise le critère suivant pour les n centres des vignettes :

$$E = \sum_{i=1}^{n} \sum_{j=1}^{2} [distance(P_{i,t+1}, N^j_{i,t+1})]^2$$
$$E = \sum_{i=1}^{n} \sum_{j=1}^{2} (N^j_{ix}.X_{i,t+1} + N^j_{iy}.Y_{i,t+1} + N^j_{iz}.Z_{i,t+1})$$

Si nous appelons $\mathcal{F}(\alpha,\beta,\gamma,u,v,w,N^j_{i,t+1},P_{i,t})$ la distance de $\tilde{P}_{i,t} = R_{\alpha,\beta,\gamma}.P_{i,t} + T_{u,v,w}$ au plan défini par $N^j_{i,t+1}$, nous pouvons écrire :

$$E = \sum_{i=1}^{n} \sum_{j=1}^{2} \mathcal{F}(\alpha,\beta,\gamma,u,v,w,N^j_{i,t+1},P_{i,t})$$

Notons $\mathbf{L} = (\alpha,\beta,\gamma,u,v,w)$ le vecteur décrivant une pose (rotations et translations). Soit $\mathbf{L}_k = (\alpha_k,\beta_k,\gamma_k,u_k,v_k,w_k)$ le vecteur décrivant la pose à l'itération k de l'algorithme de Lowe.

[3] Les plans passant par le centre optique de la caméra et respectivement par la ligne et la colonne de l'image qui caractérisent le point $P_{i,t+1}$

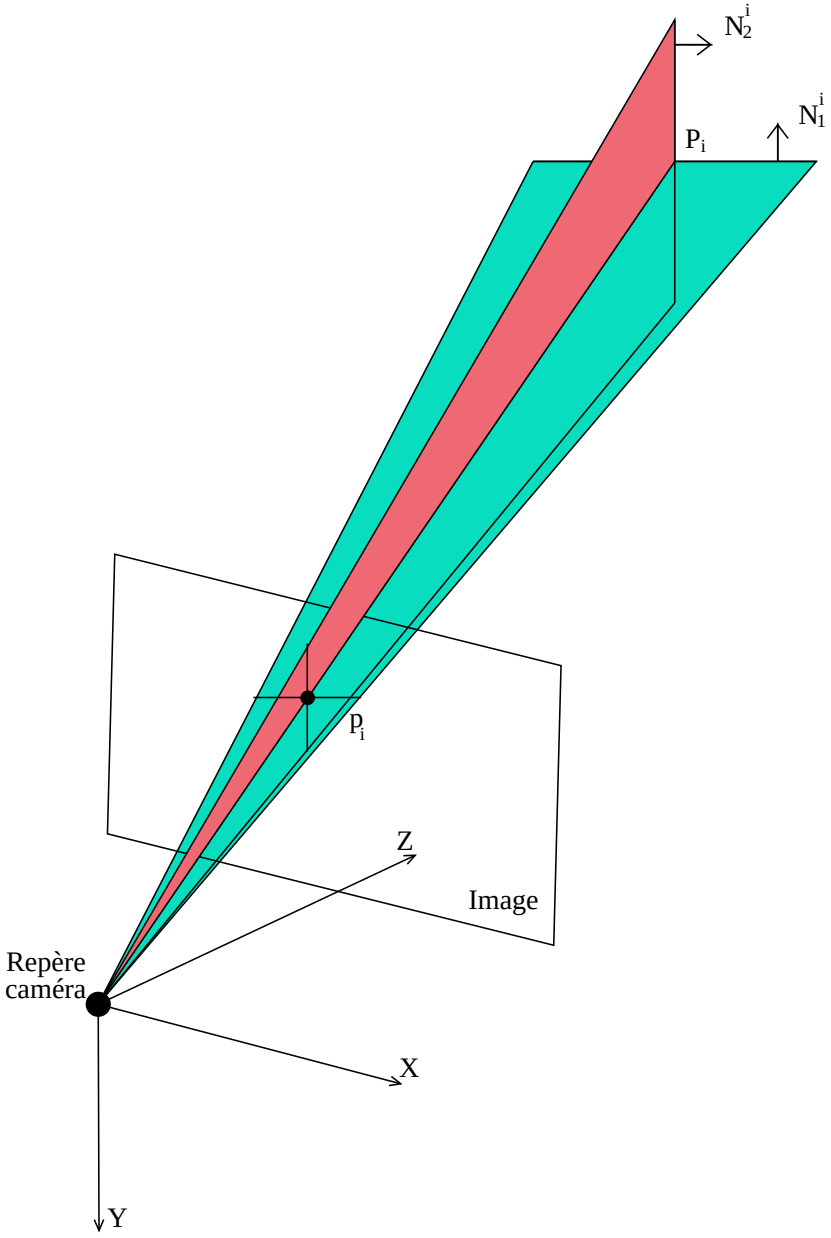


FIG. 3.6 – Algorithme itératif de Lowe : plans d'interprétation

Pour résoudre le problème avec la méthode de Newton-Raphson on effectue

un développement à l'ordre un :

$$\begin{aligned} -\mathcal{F}(\mathbf{L}_k, N^j_{i,t+1}, P_{i,t}) &= \frac{\delta\mathcal{F}(\mathbf{L}_k, N^j_{i,t+1}, P_{i,t})}{\delta\alpha}\Delta\alpha + \\ &\quad \frac{\delta\mathcal{F}(\mathbf{L}_k, N^j_{i,t+1}, P_{i,t})}{\delta\beta}\Delta\beta + \quad \dots + \\ &\quad \frac{\delta\mathcal{F}(\mathbf{L}_k, N^j_{i,t+1}, P_{i,t})}{\delta w}\Delta w \end{aligned}$$

Ou, en utilisant une notation matricielle, avec $\mathbf{F}$ un vecteur de dimension $2n$ et $\mathbf{D}$ une matrice de dimension $2n \times 6$:

$$\mathbf{F} = \begin{bmatrix} -\mathcal{F}(\mathbf{L}_k, N^1_{1,t+1}, P_{1,t}) \\ -\mathcal{F}(\mathbf{L}_k, N^2_{1,t+1}, P_{1,t}) \\ \vdots \\ -\mathcal{F}(\mathbf{L}_k, N^1_{n,t+1}, P_{n,t}) \\ -\mathcal{F}(\mathbf{L}_k, N^2_{n,t+1}, P_{n,t}) \end{bmatrix}$$

$$\mathbf{D} = \begin{pmatrix} \frac{\delta\mathcal{F}(\mathbf{L}_k, N^1_{1,t+1}, P_{1,t})}{\delta\alpha}\Delta\alpha & \dots & \frac{\delta\mathcal{F}(\mathbf{L}_k, N^1_{1,t+1}, P_{1,t})}{\delta w}\Delta w \\ \frac{\delta\mathcal{F}(\mathbf{L}_k, N^2_{1,t+1}, P_{1,t})}{\delta\alpha}\Delta\alpha & \dots & \frac{\delta\mathcal{F}(\mathbf{L}_k, N^2_{1,t+1}, P_{1,t})}{\delta w}\Delta w \\ \vdots & \vdots & \vdots \\ \frac{\delta\mathcal{F}(\mathbf{L}_k, N^1_{n,t+1}, P_{n,t})}{\delta\alpha}\Delta\alpha & \dots & \frac{\delta\mathcal{F}(\mathbf{L}_k, N^1_{n,t+1}, P_{n,t})}{\delta w}\Delta w \\ \frac{\delta\mathcal{F}(\mathbf{L}_k, N^2_{n,t+1}, P_{n,t})}{\delta\alpha}\Delta\alpha & \dots & \frac{\delta\mathcal{F}(\mathbf{L}_k, N^2_{n,t+1}, P_{n,t})}{\delta w}\Delta w \end{pmatrix}$$

On peut voir que chaque itération de l'algorithme correspond à la résolution du système matriciel suivant :

$$\mathbf{F} = \mathbf{D}.\Delta\mathbf{L} \tag{3.1}$$

Finalement, pour chaque itération, nous remettons à jour le vecteur $\mathbf{L}$:

$$\mathbf{L}_{k+1} = \mathbf{L}_k + \Delta\mathbf{L}$$

La notation $+$ est en fait un peu abusive, on construit la matrice de rotation $\mathbf{R_k}$ et le vecteur de translation $\mathbf{T_k}$ correspondant à $\Delta\mathbf{L}$, et on met à jour $\mathbf{R}_{\alpha,\beta,\gamma}$ et $\mathbf{T}_{\mathbf{u},\mathbf{v},\mathbf{w}}$:

$$\mathbf{R}_{\alpha,\beta,\gamma} = \mathbf{R_k}.\mathbf{R}_{\alpha,\beta,\gamma}$$

$$\mathbf{T}_{\mathbf{u},\mathbf{v},\mathbf{w}} = \mathbf{T}_{\mathbf{u},\mathbf{v},\mathbf{w}} + \mathbf{R_k}.\mathbf{T_k}$$

Après ce résumé de l'algorithme itératif de Lowe, il reste à introduire la robustesse dans le calcul de la pose. Nous avons pour cela ajouté un M-estimateur tel qu'expliqué par Comport *et al.* [15]. Cela consiste à ajouter un terme de pondération à l'équation 3.1 :

$$\mathbf{W}.\mathbf{F} = \mathbf{W}.\mathbf{D}.\Delta\mathbf{L}$$

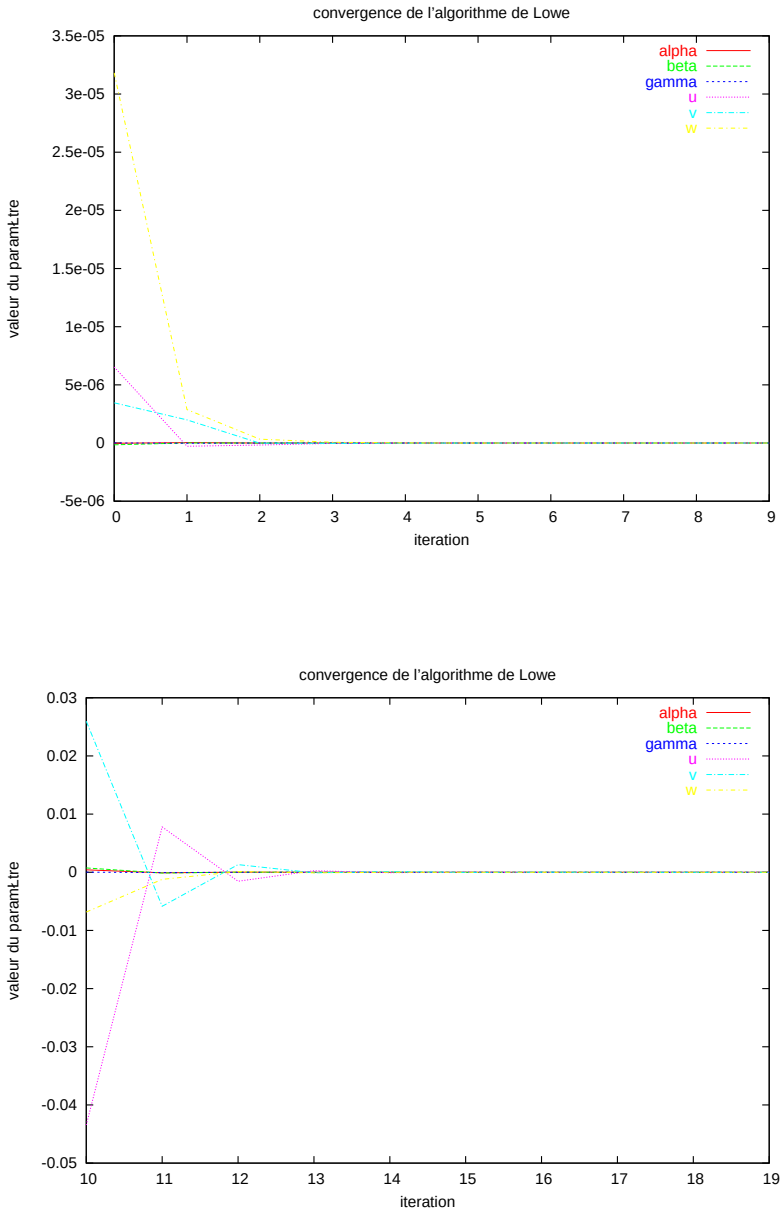


FIG. 3.7 – Convergence de l'algorithme itératif de Lowe pour quatre calculs de pose différents.

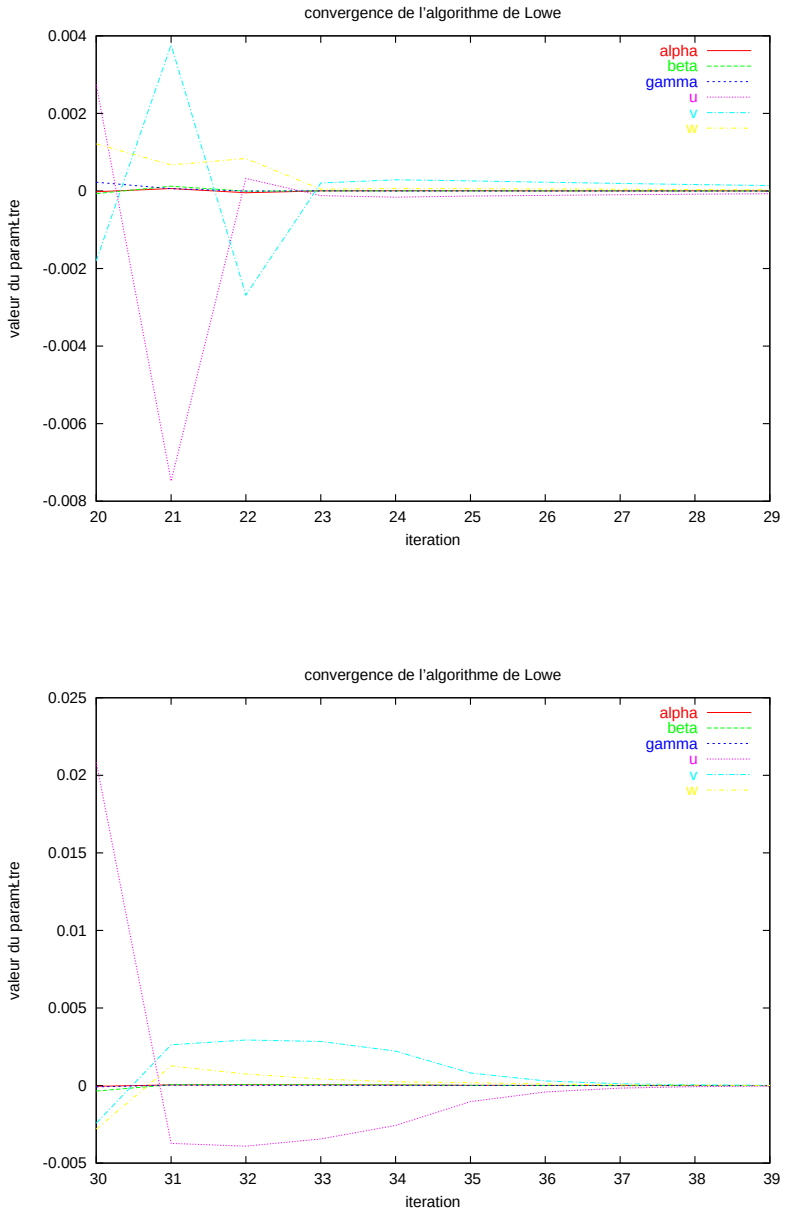


FIG. 3.8 – Convergence de l'algorithme itératif de Lowe pour quatre calculs de pose différents (suite).

avec

$$\mathbf{W} = \begin{pmatrix} w_0 & 0 & 0 & \dots & \dots & \dots \\ 0 & w_0 & 0 & \dots & \dots & \dots \\ 0 & 0 & w_1 & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & w_n & 0 \\ \vdots & \vdots & \vdots & \vdots & 0 & w_n \end{pmatrix}$$

Le calcul des w_i se fait de la façon suivante :

$$w_i = \frac{\psi(\delta_i/\sigma)}{\delta_i/\sigma}$$

où ψ est la fonction d'influence et $\delta_i = \Delta_i - \text{Med}(\Delta)$, Δ étant le vecteur des résidus. σ représente la valeur de l'écart-type du bruit sur les « bonnes » mesures, calculée par $\sigma = 1.4826$. Med($|\ \delta\ |$).

La fonction d'influence de l'estimateur de Tukey [2] est donnée par :

$$\psi(u) = \begin{cases} u(C^2 - u^2)^2 & si\ |\ u\ | \leq C \\ 0 & sinon \end{cases}$$

avec C le facteur de proportionnalité de Tukey de valeur $C = 4.6851$.

On constate qu'avec cet estimateur les appariements erronés obtiennent un poids nul, et sont donc complètement ignorés lors du calcul de pose. C'est ce qui permet à l'algorithme de supporter les occultations et de n'estimer la pose qu'à partir de correspondances de points significatives.

3.4.2 Algorithme pratique

La principale difficulté consiste dans le calcul des dérivées partielles. Pour simplifier ce calcul, à chaque étape du processus itératif, on modifie la position du modèle dans son repère de référence. On calcule les points $P_i^k = R_{\alpha,\beta,\gamma}.P_i + T_{u,v,w}$ qui deviennent les nouveaux points de référence du modèle [1]. A l'étape (k+1) on minimise alors le critère :

$$E_{k+1} = \sum_{i=1}^{n} \sum_{j=1}^{2} [\mathcal{F}(\emptyset, N_i^j, P_i^k) + \sum_{m=1}^{6} \frac{\delta \mathcal{F}}{\delta \mathbf{L}_m}(\emptyset, N_i^j, P_i^k) \Delta L_m]^2$$

Dans le nouveau repère de référence du modèle, les positions des points P_i^k sont obtenues pour $(\alpha, \beta, \gamma, u, v, w) = (0,0,0,0,0,0) = \emptyset$.

D'où $\mathbf{R}_\alpha = \mathbf{R}_\beta = \mathbf{R}_\gamma = \mathbf{Id}$

On en déduit alors les valeurs des dérivées partielles :

$$\frac{\delta \mathbf{R}_\alpha}{\delta\alpha} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \quad \frac{\delta \mathbf{R}_\beta}{\delta\beta} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \quad \frac{\delta \mathbf{R}_\gamma}{\delta\gamma} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

d'où :

$$\begin{array}{l} \frac{\delta \mathcal{F}}{\delta\alpha}(\emptyset, N_i^j, P_i^k) = N_{iz}^j * Y_i^k - N_{iy}^j Z_i^k \\ \frac{\delta \mathcal{F}}{\delta\beta}(\emptyset, N_i^j, P_i^k) = N_{ix}^j * Z_i^k - N_{iz}^j X_i^k \\ \frac{\delta \mathcal{F}}{\delta\gamma}(\emptyset, N_i^j, P_i^k) = N_{iy}^j * X_i^k - N_{ix}^j Y_i^k \\ \frac{\delta \mathcal{F}}{\delta u}(\emptyset, N_i^j, P_i^k) = N_{ix}^j \\ \frac{\delta \mathcal{F}}{\delta v}(\emptyset, N_i^j, P_i^k) = N_{iy}^j \\ \frac{\delta \mathcal{F}}{\delta w}(\emptyset, N_i^j, P_i^k) = N_{iz}^j \end{array}$$

L'algorithme se déroule alors de la façon suivante :

Calcul des normales aux plans en chaque point Pour chaque point p_i, les normales N_i^1 et N_i^2 sont calculées. Si on appelle O le centre de la caméra, ces normales sont de la forme :

$$N_i^1 = \overrightarrow{Op_i} \wedge \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad N_i^2 = \overrightarrow{Op_i} \wedge \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

Elles sont ensuite normées.

Calcul itératif de L_k Cette opération est effectuée 10 fois.

1. Calcul des résidus Δ_i, de leur médiane centrée, puis des w_i ;
2. Remplissage des matrices **F** et **D**

$$\begin{array}{l} F(2*i) = -w_i * (N_{ix}^1 X_i^k + N_{iy}^1 Y_i^k + N_{iz}^1 Z_i^k) \\ F(2*i+1) = -w_i * (N_{ix}^2 X_i^k + N_{iy}^2 Y_i^k + N_{iz}^2 Z_i^k) \\ \hline D(1,2*i) = w_i * (N_{iz}^1 * Y_i^k - N_{iy}^1 Z_i^k) \\ D(2,2*i) = w_i * (N_{ix}^1 * Z_i^k - N_{iz}^1 X_i^k) \\ D(3,2*i) = w_i * (N_{iy}^1 * X_i^k - N_{ix}^1 Y_i^k) \\ D(4,2*i) = w_i * (N_{ix}^1) \\ D(5,2*i) = w_i * (N_{iy}^1) \\ D(6,2*i) = w_i * (N_{iz}^1) \end{array}$$

et pareil pour les $D(n, 2*i+1)$, $n \in [1 \ldots 6]$.

3. Mise à jour de $R_{\alpha,\beta,\gamma}$ et $T_{u,v,w}$

La figure 3.7 montre l'évolution des valeurs des six paramètres $\alpha, \beta, \gamma, u, v, w$ pour quatre calculs de pose. On peut tout d'abord constater que l'initialisation, qui consiste à prendre comme estimation de la nouvelle pose la pose précédente, est plutôt bonne puisque les variations sont assez faibles. Ensuite on constate qu'en dix itérations, le nombre fixe d'itérations choisi, l'algorithme converge sans problème.

3.5 Résultats expérimentaux

Cet algorithme a été implanté sur un ordinateur de bureau relié à une caméra numérique SONY permettant d'obtenir un flux vidéo de 30 images par seconde sur le bus FireWire (IEEE1394).

Il a été testé sur des séquences réelles et artificielles. Pour des raisons de simplicité, un modèle cubique a été choisi. On dispose 16 patches sur chaque face du cube, chaque patch ayant sa texture décrite par un vecteur **V** de 100 éléments. L'apprentissage se fait sur 4 niveaux de perturbation (soit 4 matrices **H**, voir équation 1.1). Le niveau de perturbation le plus élevé est peu précis mais permet de tolérer des déplacements importants, le niveau le plus faible est extrêmement précis. En les utilisant l'un après l'autre, le moins précis en premier, on obtient un suivi stable et précis.

Avec ces paramètres, les deux étapes de suivi (estimation des appariements 3D-2D et calcul de la pose) demandent 20*ms* par image, l'appariement étant la phase la plus longue, approximativement 0.5*ms* par patch.

Comme on peut le voir aux figures 3.9 et 3.10, l'algorithme est robuste aux occultations : malgré les doigts qui cachent une partie de l'objet, le suivi continue.

Des séquences générées artificiellement ont été utilisées pour évaluer la précision de notre algorithme ; dans ce cas, la vérité terrain est connue. Dans ces séquences, le cube a été suivi sans, puis avec M-estimateur, et la précision des résultats a été mesurée. Comme critère de précision, nous avons utilisé la médiane des résidus dans l'image : pour chaque sommet de l'objet, la distance entre sa position reprojetée et sa véritable position dans l'image est calculée, et la valeur conservée comme critère est la médiane de ces distances.

La figure 3.11 montre la valeur de cette médiane dans le cas d'une séquence artificielle de 30 images perturbées par un bruit gaussien. Bien évidemment, sans la robustesse apportée par le M-estimateur l'objet-cible est perdu rapidement (ici à l'image 18). Avec le M-estimateur, non seulement la précision du résultat est sub-

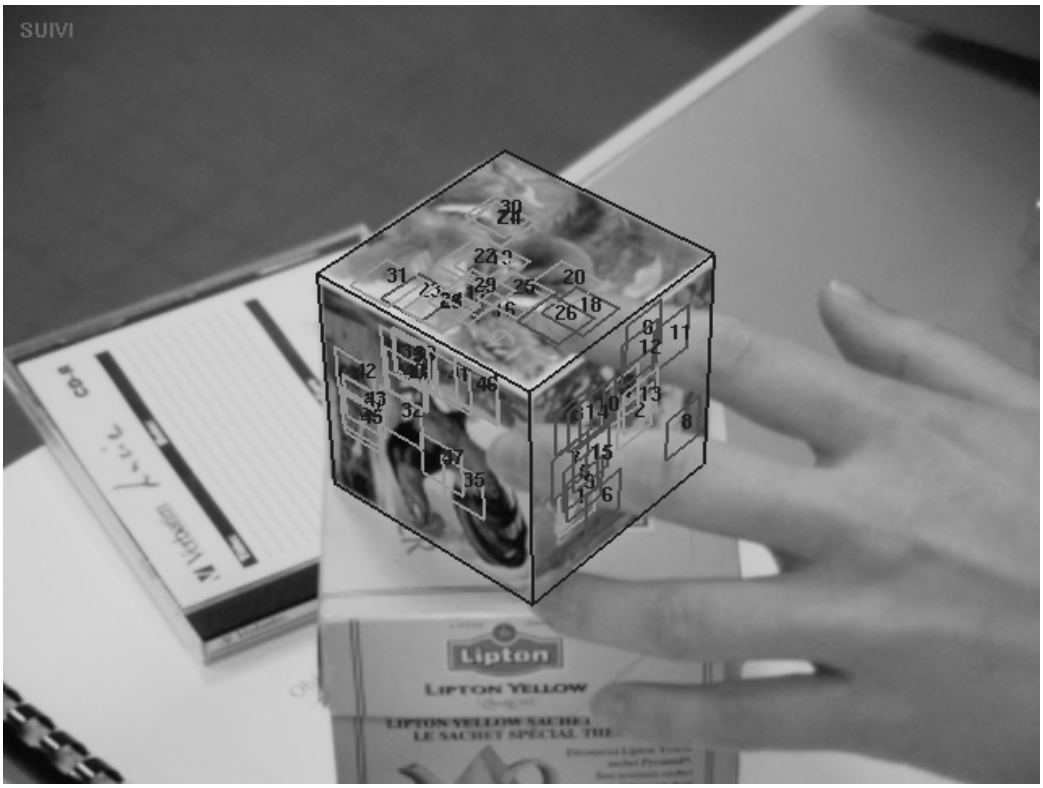


FIG. 3.9 – Suivi d'un cube occulté (1)

pixellique, c'est à dire que la distance médiane entre les sommets du cube et leur projection dans l'image est inférieure à 1 pixel, mais cette distance médiane est relativement constante au cours du temps. Cela est dû au fait que le M-estimateur ne prend pas en compte les appariements jugés mauvais.

Des extraits d'une séquence réelle sont visibles à la figure 3.13.

3.6 Conclusion et perspectives

Dans cet article nous avons décrit un algorithme efficace, précis et robuste de suivi d'objet 3D rigide.

La précision et l'efficacité sont obtenues grâce à l'utilisation d'un algorithme de suivi planaire fournissant des appariements 3D-2D avec un coût algorithmiquement faible. La robustesse est obtenue par l'utilisation d'un M-estimateur au sein d'un algorithme de calcul de pose standard.

Des améliorations pourraient être ajoutées à notre algorithme. Dans notre im-

FIG. 3.10 – Suivi d'un cube occulté (2)

plantation, la texture est apprise une fois pour toutes. On pourrait imaginer l'apprendre en ligne pour construire le modèle de l'objet plus facilement.

Une autre limitation est la zone de convergence relativement restreinte des patches de suivi : leur zone de convergence dépend de leur taille dans l'image, et comme ils sont de taille réduite leur zone de convergence est elle aussi réduite. Pour résoudre ce problème nous avons couplé notre algorithme avec un algorithme simple de reconnaissance de formes basé sur l'appariement de points d'intérêt, qui prend le relais si l'objet s'est déplacé trop vite dans l'image, et permet d'initialiser très facilement le suivi. Même s'ils sont généralement plus gourmands en temps de calcul, les algorithmes de reconnaissance ont l'avantage de pouvoir supporter n'importe quel déplacement important.

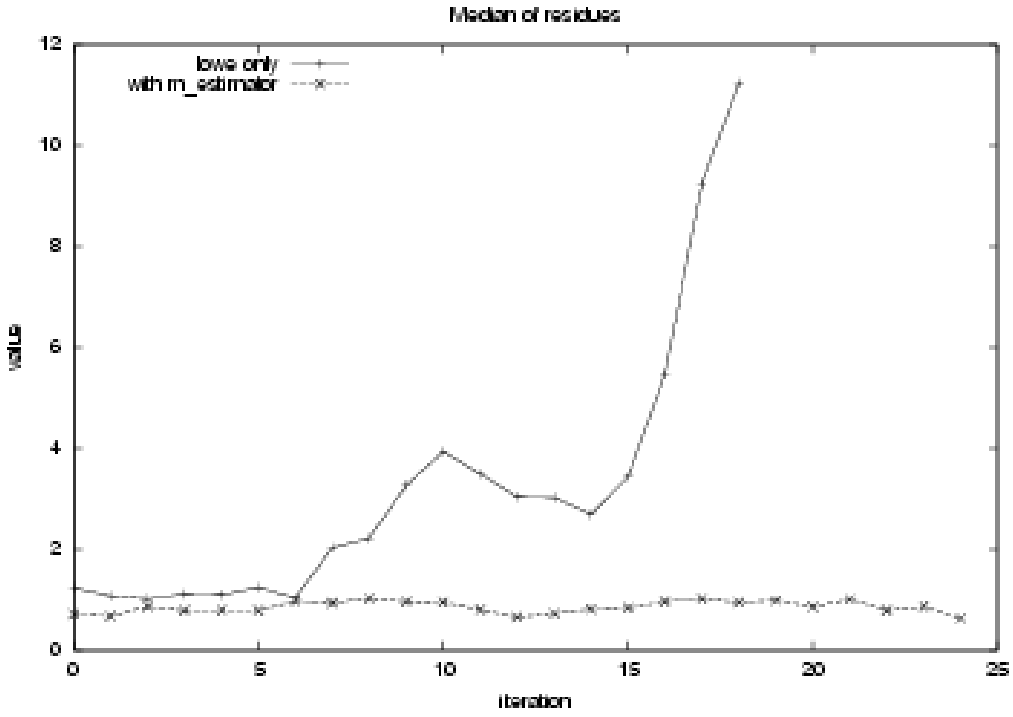


FIG. 3.11 – Médiane des résidus, en pixels, pour une séquence artificielle

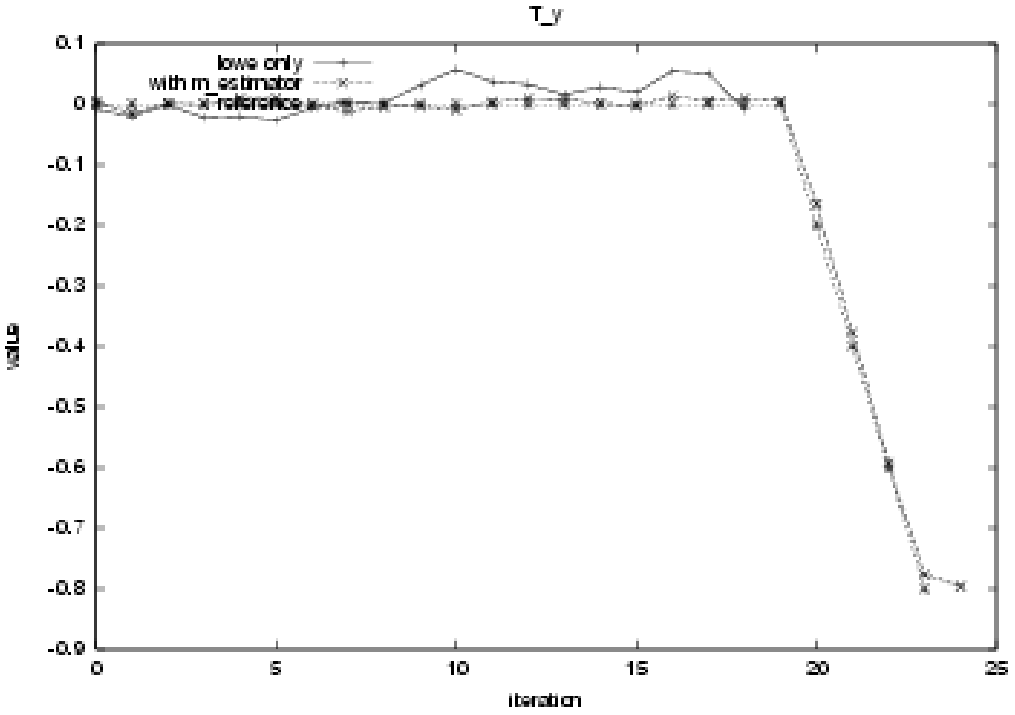
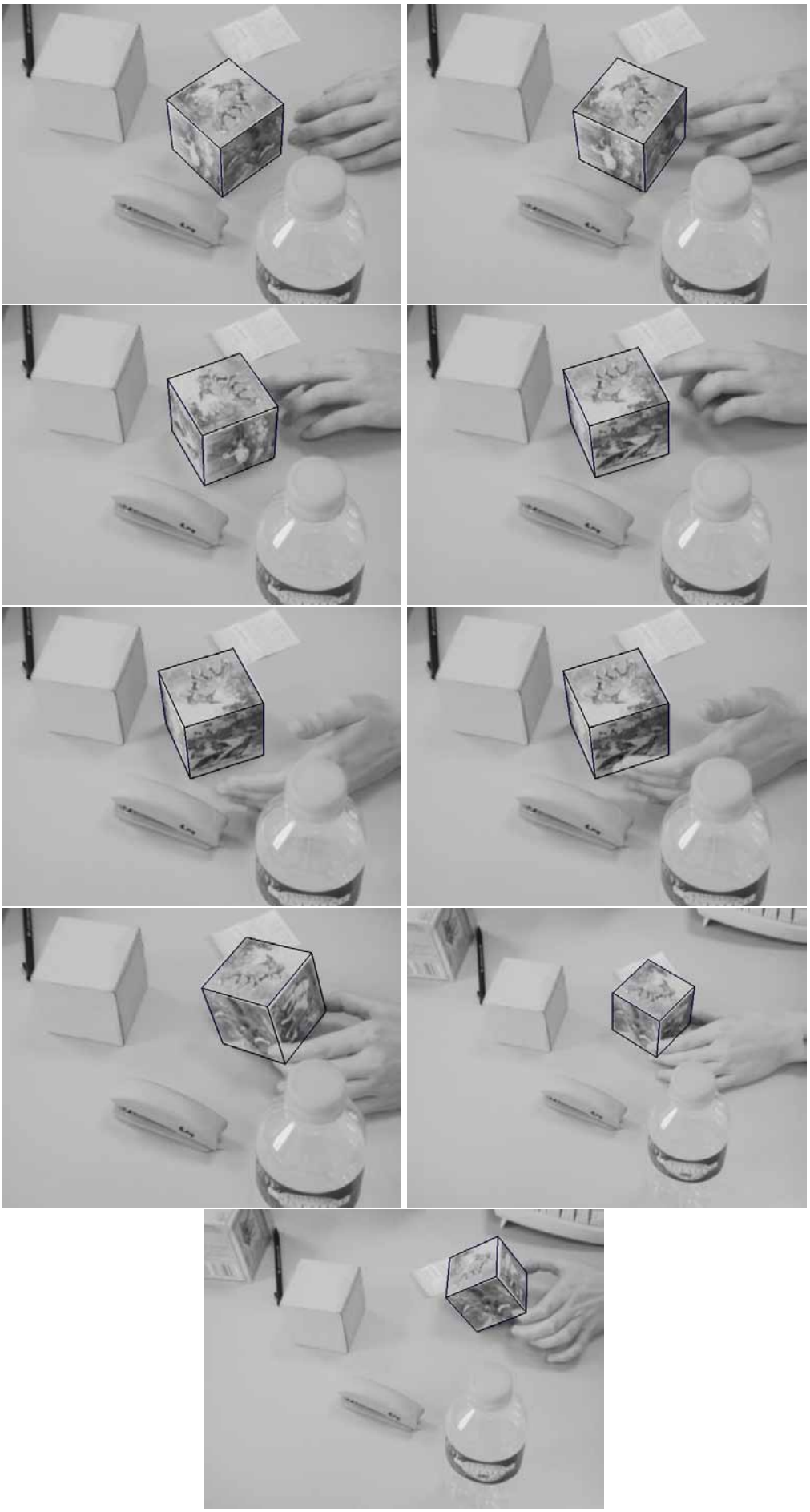


FIG. 3.12 – Estimation d'une translation sur l'axe Y



Chapitre 4

Suivi sans modèle 3D

4.1 Introduction

La méthode décrite au chapitre précédent demandait de connaître le modèle 3D de l'objet cible. C'est une hypothèse forte, et qu'il n'est pas toujours possible de vérifier quand l'objet à suivre est complexe, ou qu'il peut subir des déformations.

Les travaux suivants ont consisté à déterminer s'il était possible de se passer de ce modèle, en générant par apprentissage un modèle de l'objet. C'est de cette reflexion qu'est issu l'algorithme décrit dans ce chapitre, qui présente une méthode de suivi sans connaissance *a priori* du modèle.

L'approche proposée est basée sur l'appariement de primitives locales dans une séquence d'images. La principale différence avec les méthodes standards de ce type est que nous représentons l'apparence de l'objet 3D suivi par les déformations d'une spline multi-résolution plane de type *Thin Plate Spline* (TPS) [6].

Les splines permettent d'imposer naturellement une contrainte de lissage aux déplacements des primitives, et apportent, malgré leur nature 2D, la flexibilité nécessaire pour représenter les variations d'apparence de la projection perspective d'un objet 3D.

Dans notre approche, les relations entre les déformations 2D de la spline et les variations d'apparence liées à des changements de l'attitude spatiale de l'objet volumique d'intérêt sont capturées hors-ligne durant une phase d'apprentissage. Cette étape consiste à collecter des images représentant les divers aspects d'un objet vu sous différents angles, autour d'une vue de référence appelée dans ce mémoire « vue clef frontale ». Un appariement automatique de primitives visuelles établi des relations géométriques entre chaque image d'apprentissage et la vue clef frontale. Ces dernières servent alors de points d'ancrage pour l'estimation de la

déformation sous-jacente de la spline 2D. Un modèle statistique des déformations est ensuite calculé en réalisant une analyse en composantes principales.

Cette modélisation puissante – qui approxime les variations d'apparence dues au changement de pose 3D à l'aide de splines 2D – est ensuite utilisée pour contraindre le suivi fondé sur la texture de primitives locales.

Cet algorithme étend les capacités d'algorithmes de suivi 3D existant [1], en permettant de suivre un objet même si son modèle 3D n'est pas connu.

Avec cette méthode, il s'avère possible de suivre des visages, des objets à 2 ou 3 dimensions, des surfaces déformables. Ce type de résultats concerne de très nombreuses applications dans le domaine des loisirs, de la communication par l'image ou de la vidéo surveillance.

Ce chapitre est organisé de la manière suivante. Nous commençons tout d'abord par un rappel de l'état de l'art focalisé sur cette approche (section 4.2). Puis la section 4.3 présente les détails de notre approche et décrit ses différentes étapes. Des résultats expérimentaux sont ensuite présentés dans la section 4.4, résultats qui nous conduisent à la conclusion de ce chapitre, section4.5.

4.2 État de l'art relatif à cette méthode

Nous allons ici présenter les diverses approches apparentées à celle que nous allons décrire dans ce chapitre. Une partie d'entre elles ont déjà été présentée dans le chapitre 1, d'autres sont spécifiques à ce type de suivi et ne sont présentées qu'ici.

Notre but est de montrer les liens entre cette nouvelle approche et d'autres méthodes récentes de suivi à partir de splines 2D.

Cette approche est différente des approches de suivi basées sur des modèles, comme [23, 46, 68], qui voient le problème du suivi comme étant celui de l'estimation d'une pose à partir d'appariements de primitives de l'image (points de contours par exemple) avec des primitives du modèle 3D. La pose est alors l'ensemble des paramètres de la transformation 3D correspondante.

Cette pose est obtenue grâce à la minimisation, généralement au sens des moindres carrés, d'une fonction d'erreur. Ces méthodes donnent généralement de bons résultats mais souffrent de deux limitations : tout d'abord le modèle 3D doit être connu de façon précise, ensuite la caméra doit être parfaitement calibrée.

Les approches basées sur la texture, comme [14, 27, 38], utilisent comme primitive la totalité de la texture du motif suivi. Elles sont généralement plus sen-

[1] dans la suite on parlera de « tracker » à la place d'« algorithme de suivi », le terme anglo-saxon nous semblant exprimer de façon plus concise la notion exprimée

sibles aux occultations que les approches basées sur des modèles, même si la mise jour de certains paramètres et l'ajout de mécanismes annexes permettent d'accroître sensiblement leur robustesse (voir [34]). Ces approches sont souvent plus rapides que celles à base de primitives visuelles de par la simplicité de leurs techniques d'appariement [14].

L'idée d'utiliser des modèles déformables n'est pas nouvelle. Szeliski [66], en 1997, utilisait des *splines* pour le calcul du flot optique. Ces splines à deux dimensions servaient à représenter des champs des vecteurs de déplacement.

Belongie et Malik [6] utilisent un procédé identique pour apparier des descripteurs de forme et ainsi retrouver l'apparence d'un même objet observé sous différents angles de vue.

Ces modèles déformables peuvent aussi être utilisés pour améliorer l'ajustement obtenu à partir d'un modèle plan dans des estimations denses de flot optique [71].

Dans les approches mentionnées précédemment, le modèle déformable permet de régulariser un déplacement, mais pas d'utiliser des connaissances *a priori* sur les déformations acceptables du modèle, mise à part une contrainte de lissage par régularisation. Pour lever cette limitation, il est possible de combiner le modèle déformable avec un modèle probabiliste, afin de représenter plus fidèlement la variété des différents mouvements ou déformations possibles. Cootes *et al.* [19] génèrent ainsi un modèle en utilisant une analyse en composantes principales sur une collection d'images.

Wieghardt *et al.* [70] utilisent des appariements par graphes élastiques (*elastic graph* [42]) pour déterminer des points de correspondances d'images prises consécutivement décrivant la sphère de vues d'un objet. Les variations entre plusieurs vues du même objet sont exprimées en terme de variation des paramètres d'un espace à faible dimension, obtenu par ACP.

La méthode que nous proposons se situe entre ces différentes approches, et les améliore en se basant sur les trois points suivants :

- Suivi temps réel de primitives par une méthode inspirée de [38] ;
- Utilisation des déformations issues d'un modèle de type *Thin Plate Spline* [6] ;
- Modèle statistique déformable appris sur un jeu d'essais composé d'images typiques de l'objet observé sous différents angles.

4.3 Description de l'algorithme développé

La méthode proposée peut être décomposée en deux étapes, une phase d'apprentissage et une phase de suivi, la phase d'apprentissage étant elle-même com-

posée de deux sous-phases.

Durant la phase d'apprentissage, deux modèles différents sont appris :

- Un modèle des déformations, construit en prenant en entrée un jeu d'images représentant différentes vues de l'objet à suivre. Il s'agit d'un modèle TPS capturant les principales déformations induites par la projection perspective d'un objet 3D, observées dans différentes images et représentées par une base linéaire **Q** ;
- Des modèles de régression pour le suivi d'un ensemble de vignettes texturées, que nous appellerons « trackers ». En effet, différents trackers plans, utilisés pour le suivi de primitives visuelles locales et pertinentes, sont construits à partir de la texture située à la surface de l'objet en exploitant les images de la base d'apprentissage. Cet apprentissage donne un ensemble de modèles de régression **H** qui relient les changements de position des trackers avec les variations d'intensité lumineuse des pixels [38].

La phase de suivi utilise ces modèles pour les trois étapes suivantes, répétées à chaque image de la séquence :

1. Calcul des correspondances des primitives locales, c'est à dire production d'un ensemble de correspondances entre une image et la précédente à l'aide des trackers plans locaux ;
2. Utilisation du modèle de déformation pour optimiser la déformation du modèle global dans l'image, en accord avec le modèle des contraintes obtenu pendant la phase d'apprentissage ;
3. Calcul/prédiction des nouvelles positions des trackers plans (vignettes[2] texturés) et de la déformation du modèle global dans l'image suivante.

Nous décrivons ci-dessous plus en détail ces trois étapes.

4.3.1 Apprentissage du modèle de déformations

La première partie de notre algorithme consiste à apprendre un modèle des déformations de l'apparence de l'objet à partir d'un ensemble de vues. Ce modèle prends en compte les modifications de l'apparence de l'objet dues au changement de pose (changements par rapport à la position de référence).

Cette partie de l'algorithme est dérivée des travaux de Peters et Wieghard [57, 70], combinée avec l'utilisation du modèle *Thin Plate Spline*, appelé par la suite TPS [72].

Le modèle TPS a été préféré au graphe élastique du fait de sa simplicité ; il peut être estimé directement à partir d'un ensemble de correspondances locales

[2] ou en anglais, « patches »)

telles que des points appariés [72] ; de leur côté, les graphes élastiques utilisent un processus itératif d'optimisation plus complexe nécessitant de passer de nombreuses fois par chacun des nœuds du graphe [57, 70]. De plus, un paramètre de régularisation peut être directement inclus dans le TPS, et permet de gérer les mauvaises correspondances de points, comme nous le verrons plus loin.

Le modèle de déformation est calculé à partir d'une image clef particulière (dénommée vue frontale dans la suite de l'article), et d'une collection de plusieurs autres images capturant les changement d'apparence de l'objet d'intérêt observé depuis des points de vue voisins mais différents.

Une grille régulière est tout d'abord positionnée sur la vue clef frontale servant de référence (voir fig. 4.1). Ensuite, pour chacune des autres images de la collection, la grille est déformée de manière à minimiser la différence entre la vue frontale après déformation et l'image considérée.

Cette estimation n'est possible que si des correspondances point à point entre les deux images sont connues. Ces correspondances sont calculées en détectant les points répondant au critère de Harris [28] dans les deux images et en appariant ces points en fonction de leur descripteur SIFT [48], capturant les caractéristiques de la texture locale.

FIG. 4.1 – Déformation des splines après une légère rotation de d'une canette. haut : point d'intérêt appariés suivant SIFT - bas : estimation de la déformation de la grille

Même s'il est connu que le descripteur SIFT permet d'obtenir des correspondances relativement fiables, il n'est pas possible d'obtenir des ensembles de point homologues exempts d'erreurs (outliers). Ces erreurs affectent considérablement la minimisation par moindres carrés utilisée pour calculer la déformation des splines. C'est pourquoi le calcul de la transformation TPS est réalisé en plusieurs étapes, en utilisant un paramètre de régularisation, afin de réduire l'impact des faux appariements [72].

La *Thine Plate Spline* est une fonction à base radiale (RBF, *Radial Basis Function*), qui minimise l'énergie de courbure suivante :

$$I_f = \int\int_{\mathbf{R}^2} (f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2) dxdy$$

où $f(x,y)$ représente la nouvelle position du point (x,y) après déformation.

f est de la forme :

$$f(x,y) = a_1 + a_x x + a_y y + \sum_{i=1}^{n} w_i U(||(x_i,y_i) - (x,y)||) \tag{4.1}$$

où $U(r) = r^2 \log r$ et $U(0) = 0$. Pour que les dérivées secondes de $f(x,y)$ soient intégrables, il faut de plus que :

$$\sum_{i=1}^{n} w_i = 0 \;\; et \;\; \sum_{i=1}^{n} w_i x_i = \sum_{i=1}^{n} w_i y_i = 0$$

Dans la version régularisée de l'ajustement TPS, un paramètre β contrôle le compromis entre respect de la donnée en entrée et un lissage dans la fonction de coût :

$$K[f] = \sum_i (v_i - f(x_i,y_i))^2 + \beta I_f$$

où v_i représente la position finale après transformation. On voit que cette méthode requiert deux transformations TPS : une pour la coordonnée en x et l'autre pour la coordonnée en y, v_i correspondant à une coordonnée d'abscisse ou d'ordonnée suivant le cas.

Détermination de la TPS en fonction des appariements Revenons au calcul de la TPS d'après les informations que nous possédons. SIFT nous donne un ensemble de couples $(x_i,y_i),(u_i,v_i)$, où (u_i,v_i) sont les nouvelles coordonnées du point (x_i,y_i) dans l'image traitée, et $i = 1,2,\ldots,n$ et n le nombre d'appariements trouvés.

Déterminer la spline consiste à calculer les cœfficients $(a_1,a_x,a_y,w_1,\ldots,w_n)$ pour les coordonnées x et y. D'après les équations données précédemment, cela revient à résoudre deux fois le système linéaire suivant :

$$\left(\begin{array}{c|c} \mathbf{K} & \mathbf{P} \\ \hline \\ \mathbf{P}^t & \mathbf{0} \end{array}\right) \left(\begin{array}{c} \mathbf{w} \\ \hline \\ \mathbf{a} \end{array}\right) = \left(\begin{array}{c} \mathbf{v} \\ \hline \\ \mathbf{0} \end{array}\right)$$

avec $\mathbf{K}_{i,j} = U(||(x_i,y_i) - (x_j,y_j)||)$, v le vecteur des coordonnées u_i ou v_i suivant le cas, et $\mathbf{P}$ la matrice contenant les coordonnées des points dans l'image clef :

$$\mathbf{P} = \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \ldots & & \\ 1 & x_n & y_n \end{pmatrix}$$

```
détection points de Harris image 1
calcul histogrammes SIFT

détection points de Harris image 2
calcul histogrammes SIFT

liste ← appariements par SIFT
fixation des β_i
fixation des γ_i

calcul de la TPS à partir de (liste, β_0)
pour i de 1 à K faire
        liste_i ←appariements
                        à distance < γ_i
        calcul de la TPS à partir
                        de (liste_i, β_i)
fait
```

TAB. 4.1 – Algorithme d'ajustement TPS itératif entre deux images

Calcul itératif de la TPS Comme on l'a dit précédemment, le calcul de la spline sans régularisation risque de poser problème si il y a des outliers parmi les appariements.

C'est pour cela que l'on ajoute le cœfficient de régularisation β. Cela est fait en remplaçant la matrice $\mathbf{K}$ par $\mathbf{K}+\beta I$, où I est la matrice identité $n \times n$.

L'ajustement TPS est appliqué plusieurs fois, en commençant avec une valeur de β élevée qui diminuera progressivement au fil des itérations. A chaque étape, la spline est ajustée d'après les correspondances obtenues par le descripteur SIFT filtrées de la façon suivante : la distance entre un point clef de l'image frontale, modifié par la spline, et le point clef correspondant dans l'image cible est calculée et doit être inférieure à un seuil γ. Ce seuil diminue en même temps que β, ce qui fait que les mauvais appariements sont progressivement retirés du calcul de la spline tandis que la régularisation diminue.

Il suffit généralement de trois itérations pour obtenir un ajustement correct. Nous obtenons alors une grille déformée qui correspond à la nouvelle apparence de l'objet dans l'image traitée (voir fig. 4.1).

Construction du modèle statistique des déformations Les splines peuvent potentiellement s'ajuster à n'importe quelle déformation. Pourtant l'ensemble des déformations causées par les rotations 3D d'un objet appartiennent généralement à un sous-ensemble réduit de ces déformations possibles. Connaissant un ensemble de ces déformations possibles, nous pouvons apprendre un modèle statistique qui les explique [19].

Connaissant m de ces déformations – une déformation par image de l'objet, suivant les différents points de vue – , on les range sous forme de matrice, une ligne correspondant à une transformation. La transformation d'une spline est définie par les positions des n points de contrôle, c'est-à-dire la position des nœuds de la grille.

$$\mathbf{M} = \begin{pmatrix} x_1^1 & y_1^1 & \dots & x_n^1 & y_n^1 \\ & & \vdots & & \\ x_1^m & y_1^m & \dots & x_n^m & y_n^m \end{pmatrix} = \begin{pmatrix} \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_n \end{pmatrix}$$

où (x_j^i, y_j^i) sont les coordonnées du j^{e} nœud de la grille dans la i^{e} image de la collection.

Ces coordonnées sont connues en appliquant les équations de spline éq. 4.1 aux coordonnées des nœuds dans l'image frontale.

Une analyse en composantes principales (ACP) de $\mathbf{M}$ permet d'estimer les paramètres sous-jacents décrivant les déformations du modèle. La matrice $\mathbf{M}$ est tout d'abord centrée et réduite afin d'obtenir la matrice $\mathbf{Z}$. Cela signifie que pour chaque colonne on soustrait la moyenne des valeurs des éléments, et on divise par l'écart-type. La matrice de corrélation $\mathbf{V}$ est alors calculée par :

$$\mathbf{V} = Z^{\dagger}.Z$$

Les vecteurs propres et les valeurs propres de $\mathbf{V}$ sont alors calculés et triés par ordre décroissant.

Ces valeurs propres peuvent être séparées en deux classes ; la première, composées des valeurs propres les plus importantes, pouvant être considérées comme décrivant les changements de forme dus à des variables sous-jacentes, la seconde étant seulement due au bruit.

Pour obtenir les composantes décrivant les transformations possibles du modèle, il faut donc pouvoir séparer correctement les valeurs propres. Il existe différentes méthodes pour réaliser cette séparation. Citons en deux : le *scree-test* et la sélection des valeurs propres exprimant une part importante de la variance.

Le scree-test consiste, pour chaque valeur propre v_i, à calculer le rapport $r_i = \frac{v_i}{v_{i+1}}$. La valeur propre qui donne le plus petit r_i, r_{min}, est la valeur de séparation,

```
v        ← valeurs propres triées
nb       ← nombre de valeurs propres
r_min    ← v_0 / v_1
sep      ← 0
pour i de 1 à nb−2 faire
          r ← v_i / v_{i+1}
          si r < r_min alors
                    r_min ←r
                    sep ← i
          finsi
fait
% Séparation donnée par scree−test
```

TAB. 4.2 – Algorithme du *scree-test*

et seule les valeurs propres de rang inférieur à cette dernière sont conservées (voir tab 4.2).

La figure 4.2 et le tableau 4.3 montrent les différents résultats obtenus en fonction de la stratégie utilisée (scree-test ou choix des valeurs propres permettant d'expliquer 85% de la variance). Ce test a été effectué sur quatre jeux d'images, deux jeux de visages, un jeu portant sur une canette de boisson et le dernier sur un tapis de souris flexible.

séquence	scree-test / variance correspondante	85% variance / variance correspondante
visage 1	4 / 0.910	4 / 0.910
visage 2	3 / 0.971	2 / 0.897
tapis	6 / 0.977	3 / 0.873
canette	1 / 0.561	4 / 0.890

TAB. 4.3 – Nombre de valeurs propres conservées par scree-test et par séparation par variance

On peut voir que le scree-test est efficace : plus de 90% de la variance est expliquée par les valeurs propres conservées dans trois des quatre cas. De plus le scree-test ne nécessite pas de choisir un seuil de variance, il s'ajuste automatiquement au jeu de valeurs propres.

Pour la séquence « canette », l'objet subit un unique mouvement de rotation suivant son axe de révolution. La conservation d'un unique paramètre n'est donc pas absurde, même si la rotation n'est évidemment pas modèlisable par une unique

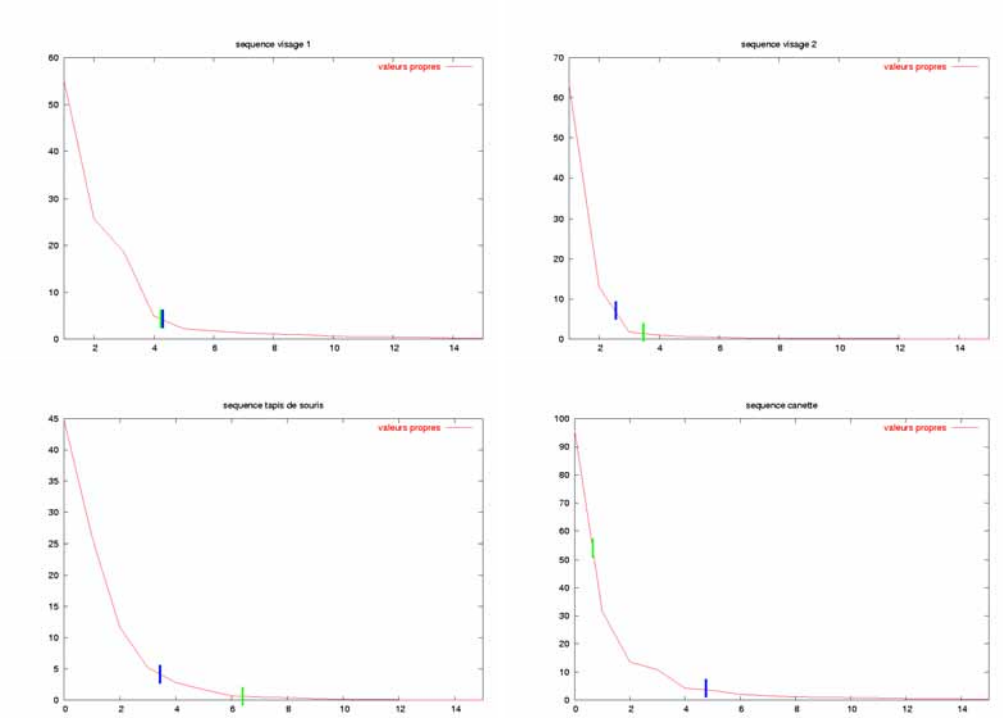


FIG. 4.2 – Différence de séparation entre le scree-test et et la conservation de 85% de la variance - marqueur bleu : séparation par variance, marqueur vert, séparation par scree-test

transformation linéaire (la plus haute valeur propre n'explique que 56% de la variance).

Une fois la séparation effectuée, et connaissant les vecteurs propres associés aux valeurs propres conservées, chaque grille peut s'écrire comme une combinaison linéaire de vecteurs :

$$\mathbf{g} = \mathbf{g_0} + a_1.\mathbf{e_1} + a_2.\mathbf{e_2} + \ldots + a_k.\mathbf{e_k} = \mathbf{g_0} + \mathbf{Q} \,.\, \mathbf{a}$$

où $\mathbf{e_i}$ est le vecteur propre correspondant à la i^{e} valeur propre et $\mathbf{a} = (a_1, \ldots, a_N)$ les paramètres du modèle (avec $N << n$) qui contrôlent la déformation.

La modélisation peut alors être décrite de la façon suivante :

$$\mathbf{g} = \mathbf{g_0} + \mathbf{Q} \,.\, \mathbf{a}$$

où $\mathbf{g_0}$ est la déformation moyenne et $\mathbf{Q}$ une matrice décrivant le modèle de déformation obtenu précédemment. Cette sorte de modélisation a été beaucoup employée en analyse de forme [19].

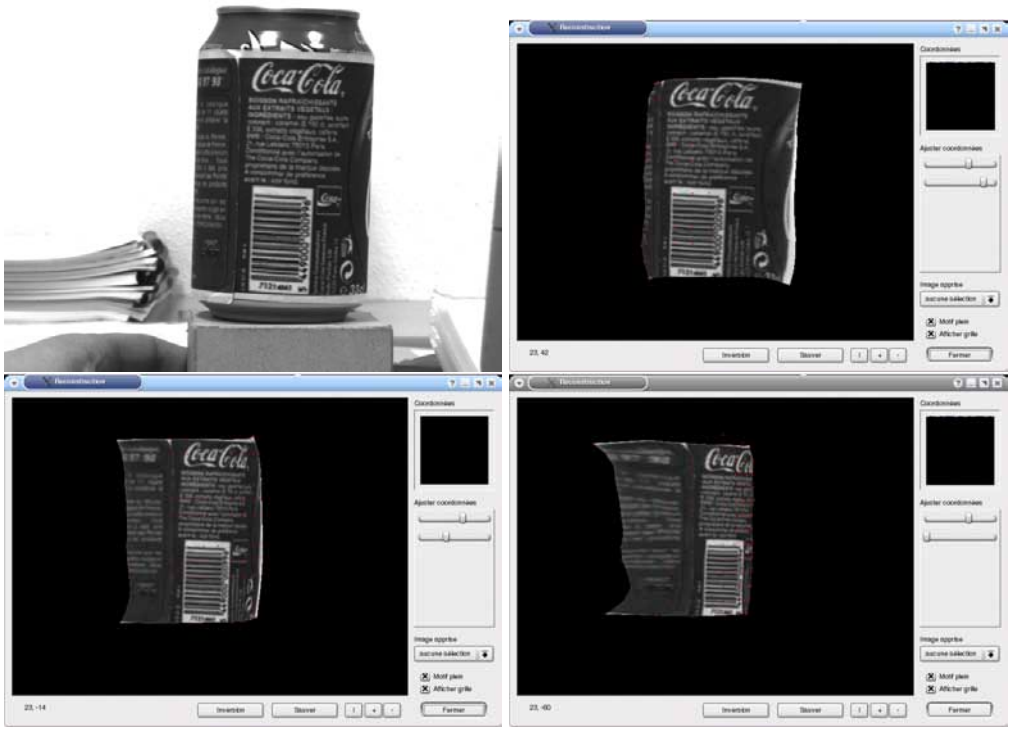


FIG. 4.3 – Modèle de canette - haut gauche : vue clef de l'objet - haut droite et bas gauche : déformation de l'image clef suivant l'une des composantes de déformation obtenue par ACP - bas droite : exagération de la déformation

Lors de nos tests avec des objets comme des canettes de soda ou des visages humains, la séparation donnait une valeur de k comprise entre deux et six.

On peut voir Figure 4.3 (en haut) comment a_j contrôle la déformation du graphe, et par conséquent l'apparence de l'objet, en accord avec les déformations apprises durant la génération du modèle. Si nous donnons une valeur de a_j hors de l'intervalle de valeurs apprises sur l'ensemble des images d'apprentissage, l'image est exagérément déformée, comme le montre la Figure 4.3 (en bas à droite).

A la fin de cette étape, nous possédons un modèle statistique des déformations, défini par une déformation moyenne $\mathbf{g_0}$ et une base linéaire $\mathbf{Q}$.

4.3.2 Apprentissage pour le suivi de petites vignettes

Dans la phase de suivi, la déformation de la grille sera contrôlée par les correspondances locales obtenues par le suivi de vignettes texturées ou « patches ». Ces dernières sont en fait des régions rectangulaires de la vue clef frontale, centrées sur les nœuds de la grille.

La Figure 4.5 montre ces vignettes sur une image de chaussure.

L'algorithme utilisé pour ces vignettes est une fois encore celui décrit partie 1.4.1.

4.3.3 Algorithme de suivi déformable

Étape 1 : correspondances locales. La première partie de cet algorithme consiste à calculer les correspondances de vignettes entre les images de la séquence. Pour chaque vignette, dans chaque nouvelle image, nous calculons les différences d'intensité $\delta\mathbf{v}$ comme expliqué précédemment. On obtient alors le vecteur $\delta\mu$ qui est une estimation des paramètres de déplacement de la vignette, en utilisant l'équation (1.1).

Connaissant ce vecteur $\delta\mu$, nous pouvons maintenant calculer la nouvelle position du centre de la vignette dans l'image courante, et par conséquent la nouvelle position du nœud de la grille qui lui correspond.

Les vignettes sont définies initialement comme étant des rectangles dans l'image frontale. Toutefois lorsque la grille est déformée leur forme est affectée par la géométrie de la grille, et l'on applique une transformation homographique à ces vignettes en appliquant le TPS calculé précédemment aux quatre coins de la vignette.

La complexité de cet algorithme est faible : seulement une multiplication matricielle et quelques opérations mathématiques simples (additions, soustractions,

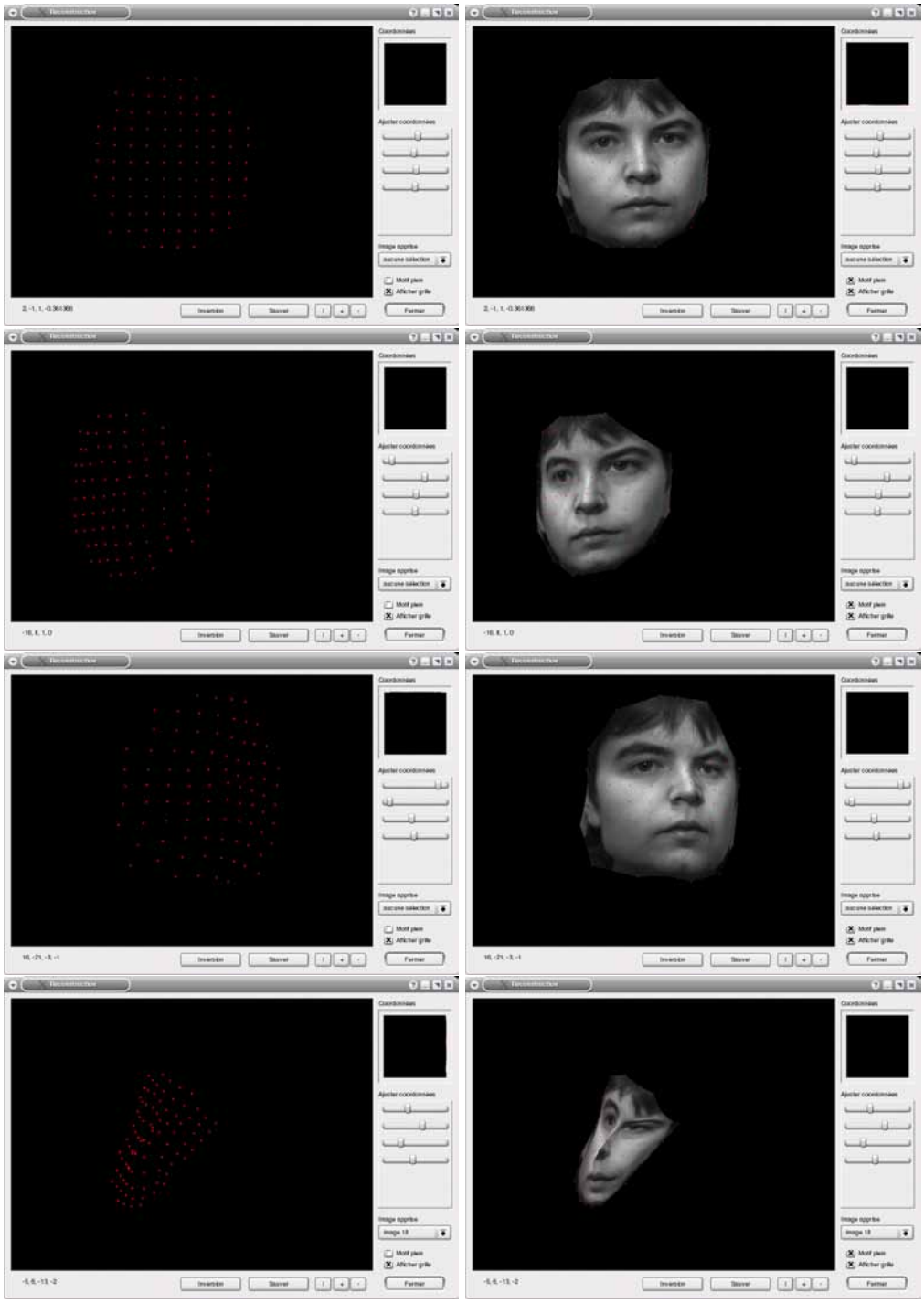


FIG. 4.4 – Modèle de visage

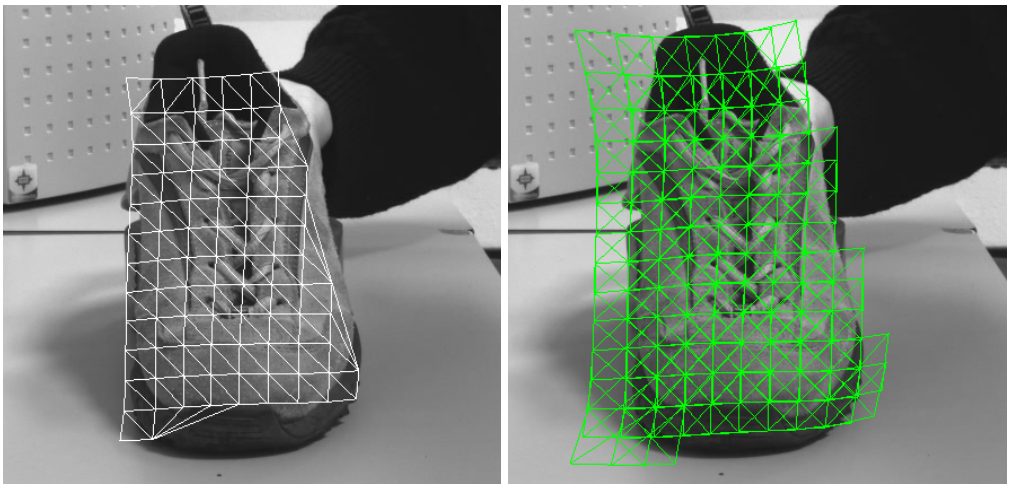


FIG. 4.5 – Patches sélectionnés sur une image de chaussure. gauche : la grille - droite : avec des vignettes sur chaque nœud

etc.) sont nécessaires pour chaque vignette. Bien entendu la complexité augmente avec le nombre de vignettes, mais il est tout de même possible d'approcher le temps réel avec 70 ou 80 vignettes sur un calculateur standard.

Étape 2 : estimation de la déformation. (algo. 4.4) Lorsque la position de chaque vignette est connue dans l'image courante, l'étape suivante consiste à déterminer la déformation du modèle, en utilisant le modèle statistique défini précédemment dans la section 4.3.1.

Comme l'optimisation à réaliser est non-linéaire, nous utilisons l'algorithme de Levenberg-Marquardt pour trouver la déformation qui minimise la différence entre les positions des nœuds et la position des vignettes correspondantes.

L'optimisation peut être formalisée comme suit. Soit $\mathbf{p_t} = (px_t^1, py_t^1, \ldots, px_t^n, py_t^n)$ l'ensemble des coordonnées des nœuds obtenues en utilisant les trackers locaux. La fonction à optimiser est la suivante :

$$O(c) = ||\mathbf{g_0} + \mathbf{Qc} - \mathbf{p}||$$

où Q est le modèle statistique appris hors-ligne à partir des images d'apprentissage. Le vecteur c contient les paramètres de déformation du modèle pour l'image courante.

```
param ← paramètres à optimiser
position_attendue ← position des noeuds de la grille,
                        donnée par les trackers
position_courante ← ACP * param
res ← residus(position_courante, position_attendue)
res_centre ← res−mediane(res_centre)
med ← mediane()
rejet ← 2.*1.4826*med

pour i allant de 0 à nombre de patches faire
        si res_centre[i] <= rejet alors
                cout[i] = res_centre[i]
        sinon
                cout[i] = 0
        finsi
fait
```

TAB. 4.4 – Fonction de coût pour l'optimisation par Levenberg-Marquardt

4.4 Résultats

Nous avons expérimenté cette méthode avec plusieurs objets tels qu'une canette de soda, une chaussure, un visage, *etc*. Cette partie se focalisera plutôt sur le suivi de visage et d'objets déformables, qui sont des problèmes sur lesquels se concentre actuellement beaucoup de recherches.

4.4.1 Suivi de visage

Le modèle du visage a été généré avec un jeu d'apprentissage de 80 images. L'image de face a été choisie comme vue initiale. Pour les 79 autres vues nous avons apparié les points d'intérêt avec le descripteur SIFT et calculé la déformation de la grille comme expliqué précédemment.

Le nombre de nœuds, pour la grille, a été sélectionné de façon à obtenir un ratio de 1 :10 entre le nombre de nœuds et le nombre de points appariés. Il est inutile d'avoir plus de nœuds que de points de correspondances, et ce ratio tient compte des faux appariements et de la répartition pas forcément uniforme des points d'intérêt trouvés par l'algorithme de Harris. De plus les vignettes liées aux nœuds situés sur les bords de la grille auront tendance à donner de mauvais résultats, il

faut donc qu'il y ait suffisamment de vignettes dont la surface repose entièrement sur l'objet suivi.

Comme le montre la figure 4.6, l'utilisation du modèle de déformation TPS sans régularisation échoue. Ce dernier ajuste la spline de façon à faire concorder chaque point détecté dans l'image frontale avec le point qui lui est apparié dans l'image cible, les faux appariements provoquent des déformations excessives de la grille. En procédant avec régularisation, itérativement, le résultat est bien meilleur. Pour cet exemple, les valeurs de β (eq. 4.3.1) étaient, dans l'ordre, (1000, 800, 300), valeurs déterminées empiriquement pour cette séquence.

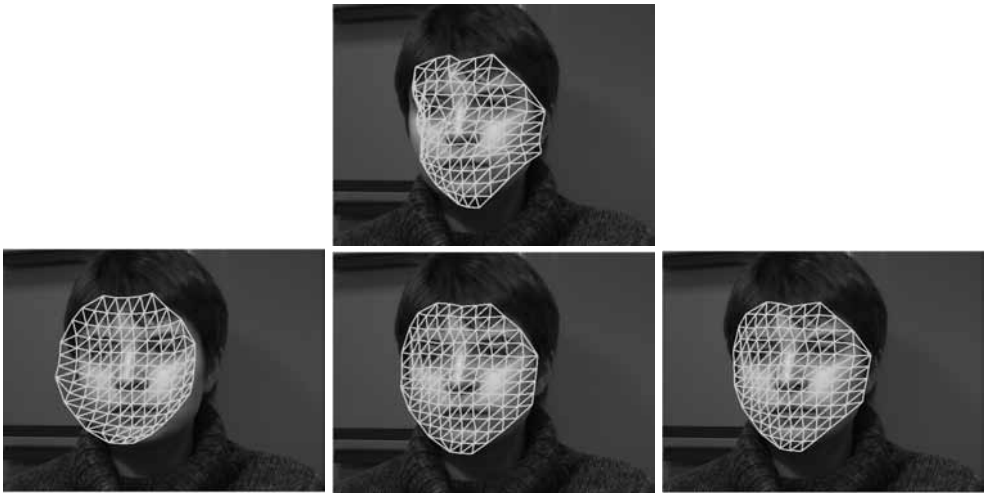


FIG. 4.6 – Ajustement de la grille par TPS. haut : sans régularisation - bas :avec une régularisation itérative

La décomposition par ACP puis le partitionnement des vecteurs propres a donné quatre paramètres sous-jacents pour ce modèle, utilisés ensuite pour suivre le visage (figure 4.7). On peut remarquer que les paramètres obtenus correspondent à peu près aux mouvements effectués : mouvements de tête de haut en bas, de gauche à droite, et inclinaison vers la gauche et vers la droite.

La figure 4.8 montre un montage réalisé à partir du suivi. Il s'agit de réalité augmentée *en ligne*, le visage de départ étant remplacé par un autre visage dont on ne connaît que l'image frontale, cette dernière étant ensuite déformée suivant les déformations du modèle capturées sur la séquence réelle.

4.4.2 Suivi d'objet déformable

Le résultat suivant, présenté à la figure 4.9, montre le suivi de surfaces déformables. Dans cette expérimentation, les vignettes étaient situées sur les 48

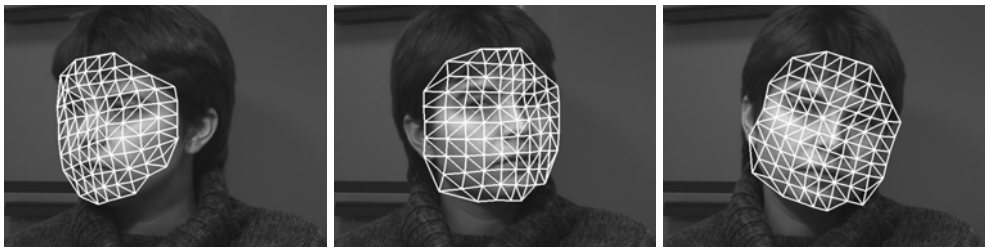


FIG. 4.7 – Suivi de visage humain

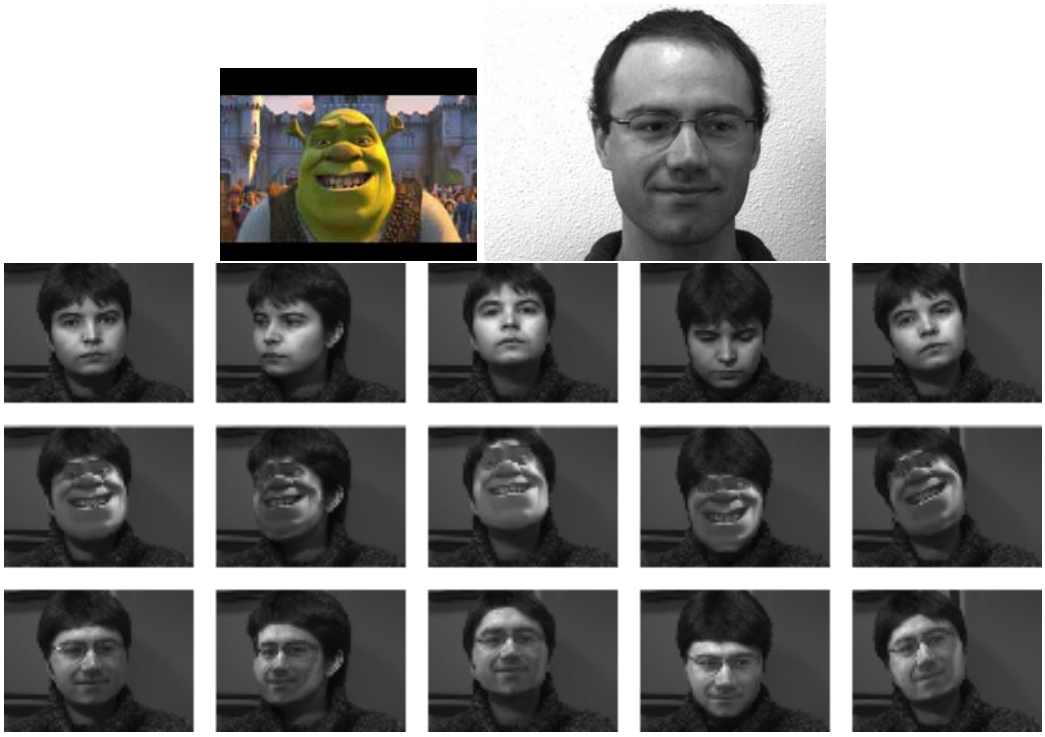


FIG. 4.8 – Substitution et déformations de visages.

nœuds d'une grille uniforme rectangulaire. Durant cette expérimentation, nous avons aussi occulté une partie de l'objet et constaté que l'algorithme continuait de converger. Sans optimisation particulière du code, le traitement de chaque image pour le suivi prends moins de 200 *ms*.

FIG. 4.9 – Suivi d'un tapis de souris « mou »

4.5 Conclusions et perspectives

Dans ce chapitre, nous avons présenté une méthode automatique pour le suivi d'objets 3D. Nous avons montré comment il est possible d'approximer les variations d'apparence induites par le changement de la pose 3D de l'objet d'intérêt par des déformations de splines 2D. Un tel modèle flexible peut-être appris automatiquement à partir d'un jeu d'images représentant l'objet sous différents points de vue.

Cette méthode est basée sur trois principes : appariement rapide de vignettes, déformation de splines TPS, et apprentissage d'un modèle statistique flexible à partir de vues typiques de l'objet.

Cette approche est facile à utiliser, ne requiert pas de modèle 3D de l'objet, et est robuste. La combinaison de mise en correspondances locales et d'un modèle statistique des déformations globales rend la méthode rapide et robuste aux occultations.

Perspectives Pour l'instant cette méthode ne fonctionne pas tout à fait en temps réel (5 images par seconde), et nous comptons l'optimiser afin qu'elle puisse traiter des séquences en temps réel vidéo. Nous allons aussi étendre cette méthode à l'utilisation de plusieurs vues clefs, car actuellement seule l'utilisation d'une unique vue clef est possible, ce qui restreint l'ampleur des rotations admissibles[3] de l'objet par rapport à la caméra.

[3] actuellement de l'ordre de 45 degrés pour la rotation de la canette de soda autour de son axe de révolution

Chapitre 5

Conclusion et perspectives

5.1 Contribution de notre travail

Le but de cette thèse était de développer des algorithmes de suivi en temps réel pour la réalité augmentée. Nous avons présenté, au fil de ce mémoire, trois techniques de suivi d'objets, dont chacune a visé à trouver un compromis entre temps de traitement et précision.

Le travail réalisé a été effectué en trois étapes, de complexité croissante :

- Suivi de surfaces planes par utilisation de composantes de texture et de contour.
- Suivi d'objets 3D simples grâce à de multiples instances d'un algorithme de suivi planaire.
- Suivi d'objets complexes par génération d'une modèle statistique de l'objet lors d'une phase d'apprentissage.

Ces algorithmes, illustrés de leurs résultats, montrent qu'il est possible de suivre des objets complexes à des vitesses tout à fait raisonnables et ainsi d'utiliser le suivi dans des applications temps réel.

Nous avons pu constater que la complexité algorithmique augmente avec la complexité des objets à suivre. Cette relation est assez évidente, mais ne doit pas être oubliée pour la conception d'algorithmes pour des applications pratiques : il faut adapter au plus juste les hypothèses sur les objets à suivre, de façon à limiter le nombre des transformations reconnues à celles effectivement possibles.

En divisant le suivi en deux phases, une phase d'apprentissage et une phase de suivi, et en accordant autant de temps que nécessaire à la phase d'apprentissage, il est possible d'obtenir des algorithmes de suivi rapides. Durant la phase d'apprentissage, les différents changements de pose possibles sont pris en compte et appris, et ainsi durant la phase de suivi seuls ces changements possibles sont

parcourus afin de trouver la position courante.

5.2 Perspectives

Les trois cas sont une adaptation d'un algorithme de suivi planaire. Si ce choix est tout à fait pertinent dans le cas du suivi de surfaces planes et d'objets contenant des surfaces planes, c'est un peu moins le cas pour les objets plus complexes.

C'est pourquoi, comme dans tout travail de thèse, il reste des pistes à explorer, par exemple concernant l'algorithme de suivi de primitives, et aussi concernant l'optimisation des algorithmes.

5.2.1 Optimisation et répartition des traitements

L'un des points à explorer concerne la vitesse d'exécution. Plus l'objet à suivre est complexe, et les données connues peu nombreuses, et plus la complexité algorithmique augmente. Heureusement, la puissance des ordinateurs augmente elle aussi, mais l'optimisation des algorithmes reste un domaine critique. En effet il ne s'agit pas seulement de pouvoir suivre en temps réel, il faut aussi utiliser ce suivi au sein d'une application. Si le suivi accapare la majorité de la puissance de calcul, il reste peu de place pour l'application qui l'utilise.

Nous avons peu parlé, dans ce mémoire, des techniques d'optimisation pour rendre les programmes informatiques plus rapides. Pour accélérer les programmes, nous avons utilisé des méthodes classiques : *profilage* pour déterminer les opérations les plus gourmandes en temps de calcul, et optimisation de celles-ci soit par amélioration de leur code, soit par réduction du nombre de leurs appels quand c'est possible. Mais il existe d'autres façons de procéder.

Une évolution intéressante est apparue dans le domaine de l'informatique grand public : désormais le processeur n'est plus la seule unité de traitement performante, et les cartes graphiques, dont le prix équivaut parfois à celui d'une carte mère et de son processeur, ont pris une part importante dans les traitements effectués.

Ces unités de traitement à part entière, entièrement dévolues au traitement d'image même si c'est pour les afficher et non les analyser, pourraient être utilisée dans les applications de suivi. Nous nous servons déjà du processeur graphique, par l'intermédiaire d'OpenGL, pour réaliser les rendus en temps réel. Mais ces cartes disposent aussi d'algorithmes de filtrage, ou de fonctions de manipulation de buffers, qui pourraient permettre de décharger le processeur de certaines de tâches de traitement bas-niveau et ainsi accélérer les algorithmes.

5.2.2 Amélioration de la technique de suivi planaire

Les deux algorithmes de suivi d'objets 3D exposés aux chapitres 3 et 4 reposent sur l'utilisation de trackers locaux. Ces trackers permettent déterminer quatre paramètres de mouvement planaires : deux translations, une rotation et un changement d'échelle. Or ils ne sont utilisés que pour connaître la nouvelle position du centre de la vignette à laquelle ils correspondent. Il y a donc une perte d'une partie des informations mesurées par ces instances de l'algorithme de suivi planaire. Il devrait être possible d'utiliser, en plus de la position du centre de la vignette, la direction de la normale à cette vignette, par exemple.

5.2.3 Remplacement de la technique de suivi planaire

L'algorithme exposé au chapitre 4 pourrait très certainement être améliorer en changeant de méthode de suivi des nœds de la grille. En effet nous avons repris pour suivre ces nœuds les instances de l'algorithme de suivi planaire utilisés au dans le suivi avec modèle décrit au chapitre 3, mais les approximations qui étaient vraies dans le cadre du suivi d'un objet composé de faces planes n'est plus vrai dans le cas d'un objet quelconque. Cette fois l'approximation qui consiste à considérer qu'au voisinage du point l'objet est plan n'est pas toujours vérifiée.

Il serait certainement intéressant d'utiliser une autre méthode de suivi, pour rendre l'algorithme plus performant, d'autant que la génération d'un modèle statistique à partir d'images-clefs donne, elle, de très bons résultats, dont on pourrait profiter plus pleinement.

Bibliographie

[1] ALLEZARD N. *Reconnaissance, localisation et suivi d'objets texturés par vision monoculaire*. PhD thesis, Université Blaise-Pascal, Clermont-Ferrand, novembre 2001.

[2] ANDREWS D., BICKEL P., HAMPEL F. *et al. Robust Estimators of Location Survey and Advances*. Priceton Univ. Press, Princeton, 1972.

[3] ARULAMPALAM S., MASKELL S., GORDON N. et CLAPP T., « A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking », *IEEE Transactions on Signal Processing*, vol. 50, n° 2, p. 174–188, février 2002.

[4] AZUMA R., BAILLOT Y., BEHRINGER R. *et al.* « Recent Advances in Augmented Reality ». cga, novembre 2001.

[5] AZUMA R., HOFF B., NEELY III H. et SARFATY R. « A Motion-Stabilized Outdoor Augmented Reality System ». dans PRESS I. C., éditeur, *IEEE Virtual Reality*, p. 252–259, Los Alamitos,Californy, 1999.

[6] BELONGIE S., MALIK J. et PUZICHA J., « Shape Matching and Object Recognition Using Shape Contexts », *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 24, n° 4, p. 509–522, avril 2002.

[7] BENHIMANE S. et MALIS E. « Real-time image-based tracking of planes using efficient second-order minimization ». dans *IEEE/RSJ International Conference on Intelligent Robots Systems*, Sendai, Japan, octobre 2004.

[8] BENHIMANE S., MALIS E., RIVES P. et AZINHEIRA J. « Vision-based Control for Car Platooning using Homography Decomposition ». dans *IEEE International Conference on Robotics and Automation*, Barcelona, Spain, avril 2005.

[9] BÉRARD F. *Vision par ordinateur pour l ?interaction homme-machine fortement couplée*. PhD thesis, Université Joseph-Fourier, Grenoble, 1999.

[10] CHALIMBAUD P., BERRY F. et MARTINET P. « The task "template tracking" in an active vision sensor ». dans *IEEE International Conference on Architecture and Machine Perception (CAMP'03)*, 2003.

[11] CHO Y., LEE J. et NEUMANN U. « A Multi-Ring Fiducial System and Intensity-Invariant Detection Method for Scaling Augmented Reality ». dans *Proc. International Workshop on Augmented Reality (IWAR)*, p. 147–165, 1998.

[12] CIPOLLA R. et BLAKE A., « Image divergence and deformation from closed curves », *International Journal of Robotics Research (IJRR)*, vol. 16, n° 1, p. 77–96, février 1997.

[13] COMANICIU D., RAMESH V. et MEER P. « Real-Time Tracking of Non-Rigid Objects using Mean Shift ». dans *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. II : 142–149, 2000.

[14] COMANICIU D., RAMESH V. et MEER P., « Kernel-based object tracking », *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 25, n° 5, p. 564–577, mai 2003.

[15] COMPORT A., PRESSIGOUT M., MARCHAND E. et CHAUMETTE F. « Une loi de commande par asservissement visuel robuste aux mesures aberrantes ». dans *14Ã¨me CongrÃ¨s Francophone AFRIF-AFIA de Reconnaissance des Formes et Intelligence Artificielle, RFIA'04*, Toulouse, France, janvier 2004.

[16] COMPORT A., KRAGIC D., MARCHAND E. et CHAUMETTE F. « Robust Real-Time Visual Tracking : Comparison, Theoretical Analysis and Performance Evaluation ». dans *IEEE Int. Conf. on Robotics and Automation, ICRA'05*, p. 2852–2857, Barcelona, Spain, April 2005.

[17] COMPORT A., MARCHAND E. et CHAUMETTE F. « Robust model-based tracking for robot vision ». dans *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'04*, vol. 1, p. 692–697, Sendai, Japan, septembre 2004.

[18] COOLEY J. et W. TUKEY J., « An algorithm for the machine calculation of complex Fourier series », *Math. Comput.*, vol. 29, p. 297–301, 1965.

[19] COOTES T., EDWARDS G. et TAYLOR C., « Active Appearance Models », *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 23, n° 6, p. 681–685, juin 2001.

[20] DEMENTHON D. et DAVIS L., « Model-based object pose in 25 lines of code », *International Journal of Computer Vision (IJCV)*, vol. 15, p. 123–141, juin 1995.

[21] DERICHE R. et FAUGERAS O., « Tracking line segments », *Image and Vision Computing (IVC)*, vol. 8, n° 4, p. 261–270, 1990.

[22] DHOME M., RICHETIN M. et LAPRESTE J.-T., « Determination of the Attitude of 3D Objects from a Single Perspective View », *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 11, n° 12, p. 1265–1278, 1989.

[23] DRUMMOND T. et CIPOLLA R., « Real-time tracking of complex structures with on-line camera calibration », *Image and Vision Computing (IVC)*, vol. 20, n° 5-6, p. 427–433, mars 2002.

[24] FALOUTSOS C., BARBER R., FLICKNER M. *et al.* « Efficient and effective querying by image content », 1994.

[25] FORSYTH D. et PONCE J. *Computer Vision : A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[26] FUHRMANN A., HESIAN G., FAURE F. et GERVAUTZ M., « Occlusion in Collaborative Augmented Environments », *Computers and graphics*, vol. 23, n° 6, p. 809–819, decembre 1999. http ://www.cg.tuwien.ac.at/research/vr/occlusion.

[27] HAGER G. et BELHUMEUR P., « Efficient Region Tracking With Parametric Models of Geometry And Illumination », *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 20, p. 1025–1039, 1998.

[28] HARRIS C. et STEPHENS M. « A combined corner and edge detector ». dans *Proceeding of the 4th Alvey Vision Conference*, p. 147–151, 1988.

[29] HUBER P. *Robust Statistics*. Wiley, 1981.

[30] ISARD M. et BLAKE A., « Condensation : conditional density propagation for visual tracking », *Int. J. Computer Vision*, 1998.

[31] JAIN A., ZHONG Y. et LAKSHMANAN S., « Object Matching Using Deformable Templates », *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 18, p. 267–278, mars 1996.

[32] JAZWINSKI A. *Stochastic Processes and Filtering Theory*. Academic Press, New York, 1970.

[33] JEPSON A., FLEET D. et EL-MARAGHI T. « Robust on-line appearance models for visual tracking ». dans *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 415–422, 2001.

[34] JEPSON A., FLEET D. et EL-MARAGHI T., « Robust Online Appearance Models for Visual Tracking », *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 25, n° 10, p. 1296–1311, octobre 2003.

[35] JURIE F. et DHOME M. « Real Time 3D Template Matching ». dans *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. (I)791–797, Hawai, decembre 2001.

[36] JURIE F. et DHOME M. « Real time template matching ». dans *Proc. IEEE International Conference on Computer Vision (ICCV)*, p. 544–549, Vancouver, Canada, juillet 2001.

[37] JURIE F. et DHOME M., « Hyperplane approximation for template matching », *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 24, n° 7, p. 996–1000, 2002.

[38] JURIE F. et DHOME M., « Real Time Tracking of 3D Objects : an Efficient and Robust Approach », *Pattern Recognition (PR)*, vol. 35, n° 2, p. 317–328, février 2002.

[39] KALMAN R. E., « A New Approach to Linear Filtering and Prediction Problems. », *Transaction of the ASME Journal of Basic Engineering*, vol. 82(Series D), p. 35–45, 1960.

[40] KOLLNIG H. et NAGEL H., « 3D Pose Estimation by Directly Matching Polyhedral Models to Gray Value Gradients », *International Journal of Computer Vision (IJCV)*, vol. 23, n° 3, p. 283–302, 1997.

[41] LA CASCIA M., SCLAROFF S. et ATHITSOS V., « Fast, Reliable Head Tracking under Varying Illumination : An Approach Based on Registration of Textured-Mapped 3D Models », *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 22, n° 4, p. 322–336, avril 2000.

[42] LADES M., VORBRUGGEN J., BUHMANN J. *et al.*, « Distortion Invariant Object Recognition in the Dynamic Link Architecture », *IEEE Transactions on Computers*, vol. 42, n° 3, p. 300–311, 1993.

[43] LEPETIT V. et BERGER M.-O. « Handling Occlusions in Augmented Reality Systems : A Semi-Automatic Method ». dans PRESS I. C., éditeur, *isar*, p. 137–146, Los Alamitos, Calif., 2000.

[44] LEPETIT V., PILET J. et FUA P. « Point Matching as a Classification Problem for Fast and Robust Object Pose Estimation ». dans *Computer Vision and Pattern Recognition (CVPR)*, 2004.

[45] LOWE D. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, 1985.

[46] LOWE D., « Fitting Parameterized Three-Dimensional Models to Images », *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 13, n° 5, p. 441–450, mai 1991.

[47] LOWE D., « Robust Model-Based Motion Tracking Through the Integration of Search and Estimation », *International Journal of Computer Vision (IJCV)*, vol. 8, n° 2, p. 113–122, 1992.

[48] LOWE D. « Object Recognition from Local Scale-Invariant Features ». dans *Proc. of the International Conference on Computer Vision ICCV, Corfu*, p. 1150–1157, 1999.

[49] LUCAS B. et KANADE T. « An Iterative Image Registration Technique with an Application to Stereo Vision ». dans *International Joint Conference on Artificial Intelligence*, p. 674–679, 1981.

[50] MARCHAND E. *"Contributions à la commande d'une caméra réelle ou virtuelle dans des mondes réels ou virtuels"*. Hdr, Université de Rennes 1, 2004.

[51] MARCHAND E., BOUTHEMY P. et CHAUMETTE F. « A 2D–3D Model-Based Approach to Real-time Visual Tracking ». dans *Image and Vision Computing*, p. 19(13) :941–955, novembre 2001.

[52] MARR D. *Vision*. W. H. Freeman and Company, N.Y., 1982.

[53] MILGRAM P. et KISHINO F., « A Taxinomy of Mixed Reality Visual Displays », *IEICE Trans. Information Systems*, vol. E77-D, no. 12, p. 1321–1329, 1994.

[54] MUKAIGAWA Y., MIHASHI S. et SHAKUNAGA T. « Photometric Image-Based Rendering for Virtual Lighting Image Synthesis ». dans *Proc. International Workshop on Augmented Reality (IWAR)*, p. 115, Washington, DC, USA, 1999. IEEE Computer Society.

[55] NEWMAN J., INGRAM D. et HOPPER A. « Augmented Reality in a Wide Area Sentient Environment ». dans *IEEE and ACM International Symposium on Augmented Reality (ISAR)*, New-York, octobre 2001.

[56] PÉREZ P., HUE C., VERMAAK J. et GANGNET M. « Color-Based Probabilistic Tracking ». dans *Proc. European Conference on Computer Vision (ECCV)*, p. 661–675, 2002.

[57] PETERS G. *A View-Based Approach to Three-Dimensional Object Perception*. PhD thesis, Bielefeld University, 2001.

[58] PRYOR H., FURNESS T. et VIIRE E. « The Virtual Retinal Display : A New Display Technology Using Scanned Laser Light ». dans *Human Factors Ergonomics Soc.*, p. 1570–1574, Santa Monica, Californy, 1998.

[59] RACICOT F.-E. et THÉORET R. « Quelques applications du filtre de Kalman en finance : estimation et prévision de la volatilité stochastique et du rapport cours-bénéfices », août 2005.

[60] SHI J. et TOMASI C. « Good Features to Track ». dans *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 593–600, 1994.

[61] SIMON G. et BERGER M.-O. « Real time registration of known or recovered multi-planar structures : application to AR ». dans *British Machine Vision Conference (BMVC)*, p. 567–576, Cardiff, U.K., septembre 2002.

[62] SIMON S., FITZGIBBON A. et ZISSERMAN A. « Markerless Tracking using Planar Structures in the Scene ». dans *IEEE and ACM International Symposium on Augmented Reality (ISAR)*, p. 120–128, Los Alamitos, Calif., 2000. IEEE CS Press.

[63] SIMONS D. et CHABRIS C., « Gorillas in Our Midst », *Perception*, vol. 28, p. 1059–1074, 1999.

[64] STAUDER J., « Augmented Reality with Automatic Illumination Control Incorporating Ellipsoidal Models », *IEEE Transactions on Multimedia*, vol. 1, n° 2, p. 136–143, 1999.

[65] SUTHERLAND I. « A Head Mounted Three-Dimensional Display ». dans AFIPS, éditeur, *Fall Joint Computer Conf.*, p. 757–764, Washington,D.C., 1968. Thompson Books.

[66] SZELISKI R. et COUGHLAN J., « Spline-Based Image Registration », *International Journal of Computer Vision (IJCV)*, vol. 22, n° 3, p. 199–218, mars 1997.

[67] TOMASI C. et KANADE T. « Detection and Tracking of Point Features », avril 1991.

[68] VACCHETTI L., LEPETIT V. et FUA P., « Stable Real-Time 3D Tracking Using Online and Offline Information », *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 26, n° 10, p. 1385–1391, octobre 2004.

[69] WELCH G. et BISHOP G. « An Introduction to the Kalman Filter ». SIGGRAPH, Course 8, août 2001.

[70] WIEGHARDT J., WURTZ R. et VON DER MALSBURG C. « Learning the Topology of Object Views ». dans *Proc. European Conference on Computer Vision (ECCV)*, p. IV : 747 ff., 2002.

[71] WILLS J. et BELONGIE S. « A Feature-Based Approach for Determining Dense Long Range Correspondences ». dans *Proc. European Conference on Computer Vision (ECCV)*, p. Vol III : 170–182, 2004.

[72] WILLS J. et BELONGIE S. « A Feature-Based Approach for Determining Long Range Optical Flow ». dans *Proc. European Conference on Computer Vision (ECCV)*, Prague, Czech Republic, mai 2004.

[73] WUNSCH P. et HIRZINGER G. « Real-Time Visual Tracking of 3D Objects with Dynamic Handling of Occlusion ». dans *IEEE Int. Conf. on Robotics and Automation (ICRA)*, p. 2868–2873, avril 1997.

[74] YAO J. et CHAM W.-K. « Efficient Model-Based Linear Head Motion Recovery from Movies. ». dans *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 414–421, 2004.

[75] ZHAO T. et NEVATIA R. « Tracking Multiple Humans in Crowded Environment. ». dans *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, p. 406–413, 2004.